NOVATESTS: METODOLOGÍA Y HERRAMIENTA SOFTWARE DE APOYO PARA LOS INGENIEROS DE PRUEBA JUNIOR

17/07/2015

PROYECTO DE FIN DE GRADO

Grado en Ingeniería de Computadores

Alumno: Adrián de Frutos Jiménez

Director: Sandra Gómez Canaval

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Resumen

La Ingeniería de Pruebas está especializada en la verificación y validación del Software, y formalmente se define como: "*Proceso de desarrollo que emplea métodos rigurosos para evaluar la corrección y calidad del producto a lo largo de todo su ciclo de vida*" [3].

Este proceso comprende un conjunto de métodos, procedimientos y técnicas formalmente definidas las cuales, usadas de forma sistemática, facilitan la identificación de la mayor cantidad de errores y fallos posibles de un software. Un software que pase un proceso riguroso de pruebas es un producto de calidad que seguramente facilitará la labor del Ingeniero de Software en la corrección de futuras incidencias, algunas de ellas generadas tras la implantación en el entorno real. Este proceso constituye un área de la Ingeniería del Software y una especialidad por tanto, de la misma.

De forma simple, la consecución de una correcta Verificación y Validación del Software requiere de algunas actividades imprescindibles como:

- Realizar un plan de pruebas del proyecto.
- Actualizar dicho plan y corregirlo en caso necesario.
- Revisar los documentos de análisis de requisitos.
- Ejecutar las pruebas en las diferentes fases del desarrollo del proyecto.
- Documentar el diseño y la ejecución de las pruebas.
- Generar documentos con los resultados y anomalías de las pruebas ya ejecutadas.

Actualmente, la Ingeniería de Pruebas no es muy reconocida como área de trabajo independiente sino más bien, un área inmersa dentro de la Ingeniería de Software. En el entorno laboral existe el perfil de Ingeniero de Pruebas, sin embargo pocos ingenieros de software tienen claro querer ser Ingenieros de Pruebas (probadores o testers) debido a que nunca han tenido la oportunidad de enfrentarse a actividades prácticas reales dentro de los centros de estudios universitarios donde cursan la carrera. Al ser un área de inherente ejercicio profesional, la parte correspondiente de la Ingeniería de Pruebas suele enfocarse desde un punto de vista teórico más que práctico. Hay muchas herramientas para la creación de pruebas y de ayuda para los ingenieros de pruebas, pero la mayoría son de pago o hechas a medida para grandes empresas que necesitan dicho software. Normalmente la gente conoce lo que es la Ingeniería de Pruebas únicamente cuando se empieza a adquirir experiencia en dicha área en el ejercicio profesional dentro de una empresa. Con lo cual, el acercamiento durante la carrera no necesariamente le ha ofrecido al profesional en Ingeniería, la oportunidad de trabajar en esta rama de la Ingeniería del Software y en algunos casos,

los recién egresados comienzan su vida profesional con algún desconocimiento en este sentido.

Es por el conjunto de estas razones, que mi intención en este proyecto es proponer una metodología y una herramienta software de apoyo a dicha metodología, para que los estudiantes de carreras de Ingeniería Software y afines, e ingenieros recién egresados con poca experiencia o ninguna en esta área (Ingenieros de Pruebas Junior), puedan poner en práctica las actividades de la Ingeniería de Pruebas dentro de un entorno lo más cercano posible al ejercicio de la labor profesional. De esta forma, podrían desarrollar las tareas propias de dicha área de una manera fácil e intuitiva, favoreciendo un mayor conocimiento y experiencia de la misma.

Abstract

The software engineering is specialized in the verification and validation of Software and it is formally defined as: “Development process which by strict methods evaluates and corrects the quality of the product along its lifecycle”.

This process contains a number of methods, procedures and techniques formally defined which used systematically make easier the identification of the highest quantity of error and failures within a Software. A software going through this rigorous process of tests will become a quality product that will help the software engineer`s work while correcting incidences. Some of them probably generated after the deployment in a real environment. This process belongs to the Software engineering and therefore it is a specialization itself.

Simplifying, the correct verification and validation of a software requires some essential activities such as:

- Create a Test Plan of the project
- Update this Test Plan and correct if necessary
- Check Requirement’s specification documents
- Execute the different tests among all the phases of the project
- Create the pertinent documentation about design and execution of these tests.
- Generate the result documents and all the possible incidences the tests could contain.

Currently, the Test engineering is not recognized as a work area but an area immerse within the Software engineering. The professional environment includes the role of Test engineer, but only a few software engineers have clear to become Test engineers (testers) because they have never had the chance to face this activities within the university study centers where they take study of this degree. Since there are little professional environments, this area is focused from a theoretical way instead of a more practical vision. There are plenty of tools helping the Test engineer, but most of them are paid tools or bespoke tools for big companies in need of this software. Usually people know what test engineering is by starting working on it and not before, when people start acquiring experience in this field within a company. Therefore, the degree studied have not approach this field of the Software engineering before and in some cases the graduated students start working without any knowledge in this area.

Because of this reasons explained, it is my intention to propose this Project: a methodology and a software tool supporting this methodology so the students of software engineering and similar ones but also graduated students with little

experience in this area (Junior Test Engineers), can afford practice in this field and get used to the activities related with the test engineering. Because of this they will be able to carry out the proper tasks of this area easier, enforcing higher and better knowledge and experience of it.

Contenido

1. Introducción

La Ingeniería de Pruebas, como área relevante dentro de la Ingeniería de Software, es aún un área que demanda profesionales y que aún tiene mucho por ofrecer acorde a la evolución tecnológica. Tradicionalmente, las pruebas siempre han quedado relegadas a un segundo plano dentro de un proyecto de desarrollo software, dándole a veces mucha menos importancia de la que en realidad deberían tener puesto que adquieren un papel más que secundario. Pero actualmente esto está cambiando. Con la creación de nuevas ISOs y la preocupación cada vez mayor de desarrollar un software de calidad y maduro para ser competente con el mercado y los tiempos cada vez más exigentes, la Ingeniería de pruebas está convirtiéndose en un valor al alza, llegando a compararse ahora mismo al nivel que el propio desarrollo del software o incluso, a la especificación de requisitos. Con la aparición y la evolución de las metodologías ágiles, uno podría decir que el proceso software ha dejado de ser un proceso a escala piramidal, donde uno tiene mayor importancia que el resto, para ser más bien una figura lineal, donde todos están al mismo nivel, y deben trabajar codo con codo.

1.1 Justificación

Como se ha mencionado anteriormente, dado que existen carencias de herramientas básicas y didácticas para el entrenamiento de los ingenieros de prueba desde el ámbito académico, es interesante plantearse la necesidad de que exista una herramienta de utilidad práctica para ayudar a suplir esta carencia. Para que realmente esta herramienta pueda ser aprovechada por los estudiantes para obtener los conocimientos prácticos necesarios, es importante acompañarla de una metodología que guíe a los estudiantes en el seguimiento sistemático de las fases de la Ingeniería de Pruebas. En este contexto, es en el que este proyecto tiene sentido y contribuye en el acercamiento de los estudiantes hacia los entornos profesionales: una metodología práctica de aplicación de la Ingeniería de Prueba en desarrollos software y una herramienta que acompañe dicho seguimiento.

Se observa, que tradicionalmente el enfoque de las asignaturas del área de Ingeniería de Software suelen seguir el proceso de desarrollo de software clásico o si siguen otras metodologías de desarrollo de software más recientes como la Programación Extrema o el desarrollo conducido por pruebas, no siempre se alcanzan un nivel práctico cercano a la realidad que las empresas requieren de los Ingenieros Software en el ámbito profesional. En este contexto, las actividades prácticas de la ingeniería de pruebas parece que o bien se abordan de forma independiente y teórica en una asignatura, se abordan por partes bien diferenciadas en asignaturas con una visión más global de la ingeniería del software o no se abordan en un contexto práctico real. En cualquiera de estas perspectivas, se requiere de herramientas de apoyo que abarque o aúne todos los conocimientos que el estudiante va adquiriendo y de una metodología que permita utilizar dichas herramientas de forma sistemática. Por esto es que remarco mi propuesta de crear una herramienta (y su metodología) que compacte todo el proceso anteriormente mencionado y facilite la aplicación real de los conocimientos adquiridos en las asignaturas que enseñen el proceso

desde un punto práctico y con visión de conjunto. Además, como se ha mencionado en el Apartado 3.4, hay muy pocas herramientas de este tipo ya que la mayoría abarcan cada área de la ingeniería de pruebas desde un subgrupo bien diferenciado, habiendo encontrado únicamente una que abarque todo (ver Tabla 1). Hay más herramientas que componen el proceso entero de la Ingeniería de pruebas, no obstante son peticiones por encargo de empresas y en su mayoría, no se dan a conocer de cara al público por motivos de copyright.

Respecto de los aspectos metodológicos del desarrollo de este proyecto de fin de grado, en el capítulo siguiente se explicarán y fundamentarán las diversas opciones y procedimientos que se han acordado para la creación y desarrollo del software que da lugar a este proyecto, atendiendo a principios de usabilidad e implementación de los mismos criterios que son objetivos de este proyecto desde un marco formal.

1.2 Objetivos Generales

Desarrollar una metodología y una herramienta software de acompañamiento para el aprendizaje y desarrollo de los procesos de la Ingeniería de Pruebas basándose en la experiencia y el ejercicio profesional en entornos de trabajo reales. Darlas además a conocer, en entornos educativos y a ingenieros que quieran introducirse en este campo de la Ingeniería Software de una forma básica y sencilla.

1.3 Objetivos Específicos

- Diseñar una metodología que defina los procesos de la Ingeniería de Pruebas que facilite las actividades de creación y diseño de las pruebas, su configuración, ejecución y corrección, la gestión de fallos e incidencias y la documentación.
- Evaluar los requisitos necesarios para la correcta funcionalidad del software y sus componentes.
- Especificar, generar y registrar dichos requisitos previamente concertados con el cliente en un documento con una plantilla completa y adecuada.
- Implementar una herramienta software de soporte a la metodología propuesta en este proyecto, que permita la automatización, gestión con criterios de usabilidad y mejora en la eficiencia de los resultados de las actividades relacionadas con la Ingeniería de Pruebas.
- Probar y verificar que el software tiene su correcto funcionamiento y valida los requisitos especificados y el fin para el que fue implementado de una forma correcta y sin errores.

- Ofrecer a la comunidad universitaria y académica, así como también a los Ingenieros de Pruebas Junior o Ingenieros recién egresados en este campo de la Ingeniería del Software, de una metodología y una herramienta software que sirva de apoyo para el aprendizaje y la práctica de las actividades propias derivadas del procesos realizados en entornos o casos de uso reales.

1.4 Estructura del documento

En este contexto, este proyecto de fin de grado consta de varios capítulos. En primer lugar, se presenta un resumen del el marco teórico que sustenta este proyecto, posteriormente se presenta el capítulo de Trabajo Relacionado donde se presenta un análisis de las herramientas existentes para desarrollar actividades propias de la Ingeniería de Pruebas. A continuación se introduce el capítulo de las contribuciones de este proyecto de fin de grado se presentan a saber: Capítulo 4: Metodología, Capítulos 5 y 6: Análisis y diseño, e implementación de la herramienta software NovaTest, respectivamente, y finalmente se representan las resultados y conclusiones de este Proyecto de Fin de Grado.

2. Marco Teórico

2.1 La Ingeniería de Pruebas dentro de la Ingeniería de Software

La ingeniería de pruebas es un área dentro de la ingeniería del software que cada vez está generando mayores demandas de profesionales en el mundo de las tecnologías de la información. Asegurar la calidad del software, es algo cada vez más importante y complejo no solo desde el punto de vista del cliente que necesita un software a medida sino desde el punto de vista del equipo de desarrollo. Para obtener resultados satisfactorios en términos de la calidad de software, las pruebas deben adecuarse a modelos y estándares que garanticen la entrega de un software probado en todos los niveles. Además, el uso de herramientas se hace cada vez más necesario para la organización, automatización y desarrollo de las pruebas puesto que en la actualidad cada vez se apuesta más por soluciones modulares, independientes, distribuidas en diferentes plataformas y entorno.

Las Fases en términos generales, de la ingeniería de software [13]:

1. **Diseño de requisitos**: durante esta etapa del desarrollo se especificará que hará detallada y específicamente, el programa a desarrollar de forma atómica. Es decir, que se requiere del software, a todos los niveles: nivel técnico, operacional y funcional. Explicado de tal manera que no quede ninguna duda y dividido de tal forma que sea sencillo de analizar, comprender e implementar más adelante.

2. **Diseño**: Durante esta fase el cliente y productor, se pondrán de acuerdo en términos de la apariencia y, valga la redundancia, el diseño que el software tendrá así como ideas de alto nivel de abstracción y otros elementos similares.

3. **Implementación**: Fase en la cual el software empieza a ser desarrollado. Esta etapa puede alargarse y se puede volver a ella en diferentes fases de un proyecto generando lo que serían distintas versiones de un mismo software hasta que éste se considere maduro y satisfactorio. E.G: prototipo, versión 1.0, versión 2.0, alfas y betas… Normalmente la transición de vuelta a esta etapa ocurre tras la validación, en la cual se han encontrado errores de implementación o diseño.

4. **Verificación y Validación**: Esta es la fase que más nos concierne de cara al desarrollo de este proyecto. Durante esta fase se desarrollaran y ejecutaran las pruebas al software. Cada una de estas pruebas, por lo general, intentarán cubrir uno o más requisitos mediante el uso del software y pueden ser clasificadas en diferentes grupos como se explicará más adelante (operacionales, de máxima capacidad, de estabilidad o unitarias). Cada vez que en esta fase no se consiga un resultado, o se llegue a un margen establecido de

satisfacción del software, una nueva versión deberá ser desarrollado iterando de nuevo a la fase anterior del proceso.

5. Mantenimiento: Esta es la fase final de un proyecto y normalmente se refiere a garantizar un porcentaje de disponibilidad de uso. Durante esta fase cualquier problema que el cliente pudiera tener será analizado y reparado en la mayor brevedad posible para satisfacer el acuerdo de fiabilidad o confianza del software.

Las metodologías de desarrollo software intentan brindar una guía de cómo abordar cada una de las fases que se han mencionado anteriormente. Las metodologías de desarrollo software más reciente, exigen que la Ingeniería de Pruebas esté contemplada a los largo del ciclo de vida del desarrollo del software. A continuación se presenta un resumen de cada una de las metodologías de desarrollo de software más relevantes en la literatura resaltando cómo incorpora la Ingeniería de Pruebas en ellas.

2.1.1 Metodología del desarrollo clásico de software

El método por excelencia para el desarrollo de software se conoce como el método clásico. Lo es también su posterior extensión conocido como es el método de cascada (waterfall en inglés). Es un modelo secuencial (o también conocido como lineal), muy simple de entender y de usar. En el desarrollo en cascada, cada fase del ciclo de vida debe ser completada para que la siguiente pueda dar comienzo. Es por esto que es recomendable para proyectos pequeños sin gran cantidad de requisitos [1].

Cada final de fase cuenta con una evaluación para comprobar que el proyecto está encaminado hacia la orientación y los deseos del cliente y la empresa. En caso negativo, el proyecto será desechado y uno nuevo puede ser planificado, o una fase puede ser rediseñada.

En este método, la fase de Ingeniería de pruebas (que es la que nos enfocamos en este proyecto) no se desarrolla hasta que el producto software no haya sido completado. En el modelo en cascada las fases nunca se solapan, de modo que, como ya hemos mencionado, una no comienza sin terminar la otra completamente [1].

La siguiente ilustración muestra de forma gráfica, la metodología de cascada:

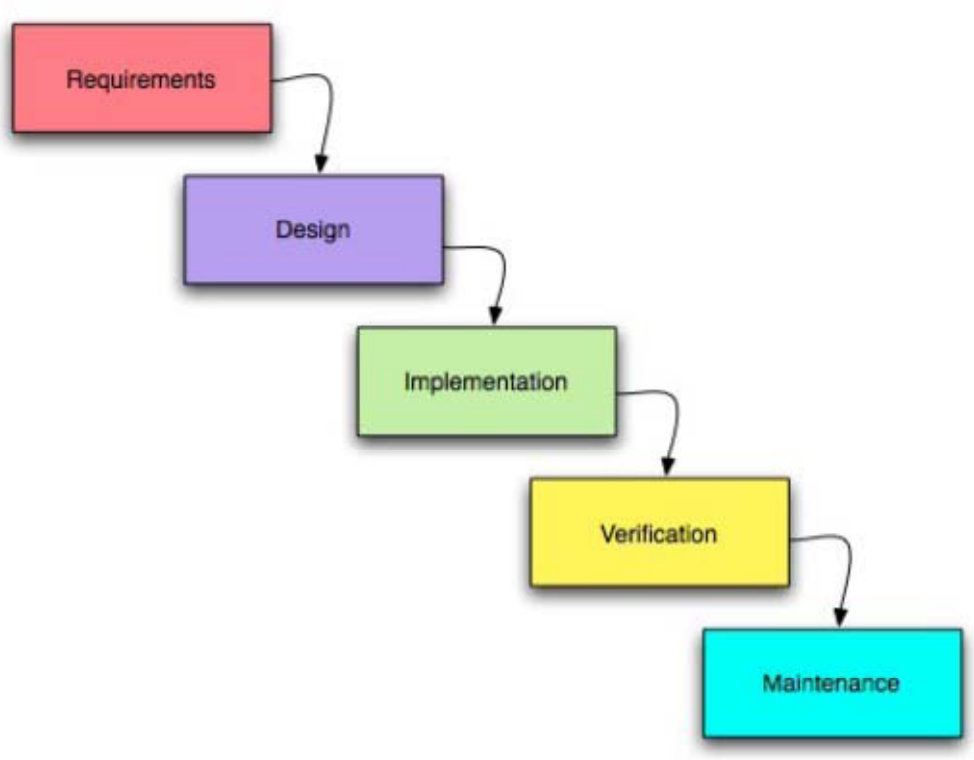

Figura 1: Fases del ciclo de vida de software del modelo en cascada

Ventajas de la metodología en cascada:

- Fácil de entender y de usar.
- Fácil de administrar debido a las fases de evaluación con objetivos específicos.
- Las fases no se solapan, de modo que solo te centras tus esfuerzos en completar una fase cada vez.
- Útil para proyectos pequeños y con pocos requisitos.

Desventajas del método en cascada:

- Una vez se entra en fase de pruebas es difícil volver hacia atrás y cambiar cualquier cosa que no haya sido previamente concebida.
- El software se desarrolla en una fase muy tardía del ciclo de vida.
- Gran cantidad de riesgos.
- No recomendable para proyectos grandes.
- No recomendable para proyectos con requisitos con probabilidades altas de ser cambiados.

Existe un modelo de ciclo de vida que incluye mejoras sobre el modelo clásico secuencial: ***el modelo en V***, explicado a continuación.

2.1.2 Modelo V del desarrollo de software

Este tipo de metodología también es secuencial pero las pruebas del producto se realizan en paralelo junto a cada fase correspondiente del desarrollo, como se puede observar en la Figura 2 [1].

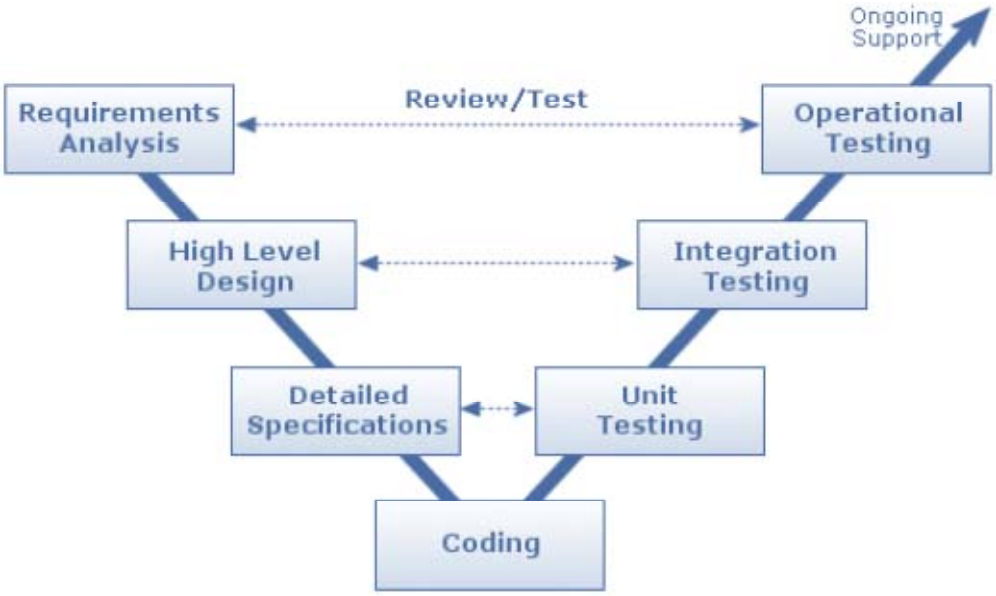

Figura 2: Fases del ciclo de vida de software del modelo en V

Durante la fase de análisis de requisitos, previo al desarrollo del software, se crea un plan de pruebas del sistema. Este documento se centra en que la funcionalidad del software coincida con los requisitos establecidos durante esta fase previa.

El diseño de alto nivel se centra en arquitectura y diseño del software. Para este caso las pruebas que se desarrollan para que cada pieza del sistema sea capaz de trabajar en conjunto. Haciendo del sistema un único componente.

En las especificaciones detalladas, cada componente del software es desarrollado definiendo la lógica del sistema. En este caso se crean planes de prueba para cada uno de los componentes por separado, para que puedan ser validados individualmente.

Por último, la fase de implementación codifica lo generado en las fases previas. Al finalizar estas fases se crea un plan de pruebas unitarias.

2.1.3 Metodología Iterativa / incremental

A diferencia de un modelo secuencial, el modelo iterativo no comienza con una especificación completa y detallada de los requisitos, sin embargo, el desarrollo comienza con un componente del software. Al terminarlo, este componente es evaluado para

comprobar su correcto funcionamiento y elaborar una futura especificación de requisitos para el componente siguiente (que es la integración del anterior con el siguiente). Este proceso es repetido una y otra vez produciendo nuevas versiones de software cada vez hasta que el resultado obtenido es el deseado. Al final del ciclo de vida de esta metodología, hay una revisión y evaluación global para comprobar que todos los componentes del sistema funcionan de forma adecuada [1].

En la Figura 3 y la Figura 4, se muestra un esquema de cómo debería ser una metodología incremental.

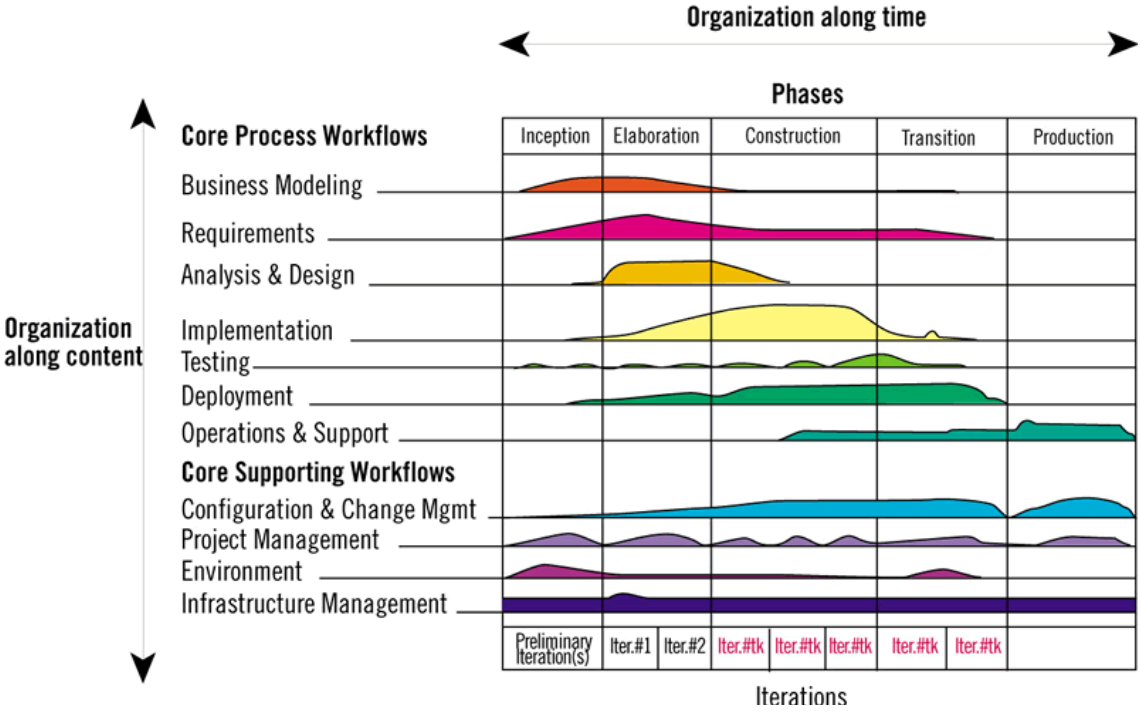

Figura 3: Fases del ciclo de vida de software iterativo

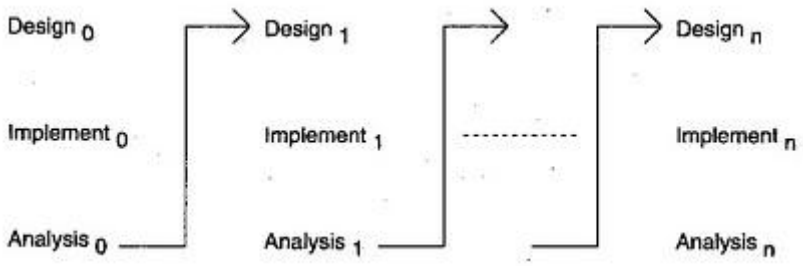

Figura 4: Iteración de actividades dentro del modelo iterativo

Ventajas del modelo incremental:

- El diseño no necesita ser especificado y desarrollado de un solo golpe, se puede crear un esqueleto que vaya evolucionando a medida que avanza el proyecto, añadiendo funcionalidades a medida que se desarrolla.
- Se pueden encontrar defectos en fases iniciales haciendo más sencillo el desarrollo en fases posteriores, al empezar con un software muy pequeñito y controlado.

- Puedes contar con opiniones y evaluaciones del cliente en cada versión del producto, de este modo tiene una visión más precisa de lo que se está desarrollando y de si esa es la visión que quiere del producto.
- Se necesita menos tiempo para la documentación.
- Es recomendable para proyectos pequeños o medianos.

Desventajas del modelo incremental:

- Las fases no se solapan y por tanto cada fase debe terminar para comenzar la siguiente, esto puede ocasionar que el proyecto se alargue en el tiempo.
- Pueden surgir problemas graves de diseño o arquitectura al no estar todos los requisitos especificados desde un principio.

2.1.4 Modelo en Espiral

El modelo en espiral es un tipo de modelo incremental. La diferencia que tiene con este modelo, es su profundo análisis en los riesgos. Para la realización de la Ingeniería de pruebas usando este modelo, cabe a destacar cuatro fases principales, a saber: Planificación, Análisis de riesgos, Ingeniería y Evaluación [1].

La planificación consiste en reunir los requisitos necesarios en los que usar la metodología en espiral. Estos requisitos serán los que más adelante consigan determinar el software a probar como satisfactorio si todos ellos se consiguieron validar durante las pruebas, o no satisfactorio si algunos de estos requisitos no se consiguen validar. Esta fase es de suma importancia, ya que no solo es el punto de partida de la metodología de pruebas en espiral sino que además, una mala elección de requisitos que son necesarios y el entendimiento de los mismos, pueden suponer una diferencia abismal entre el éxito y el fracaso.

El análisis de riesgos es un proceso que identifica los riesgos a priori que un proyecto pueda tener y sus posibles soluciones más comunes. Tras la fase de análisis de riesgos se crea un prototipo con el cual se hará un primer análisis más profundo. Si durante dicha fase se encuentran más riesgos potenciales, entonces vuelven a ser analizados para encontrar soluciones alternativas. Con esta fase se pretende limpiar, antes de que el propio software este desarrollado, la mayor cantidad de fallos genéricos y más comunes acorde al software validado. Y posteriormente, seguir con una dinámica para la solución de los riesgos que vayan surgiendo más adelante. Esta fase reduce en parte el trabajo que se realiza más adelante ya que cuantos más fallos se limpien más fluidez de validación tendrá el software.

Durante la fase de Ingeniería, el software se desarrolla y posteriormente se entra en fase de pruebas. Durante esta fase se limpian todos los fallos que hayan podido quedar en el software desarrollado. Esta es la tarea principal de un Ingeniero de Pruebas, el tema que nos atañe. Una vez se consiguen probar los requisitos escogidos en la fase primera, sin que

ninguno de ellos o al menos un porcentaje establecido por parte del cliente y la empresa, sean validados de una forma correcta. Con esto nos queremos referir a que durante la ejecución de la prueba asociada al requisito, ninguna anomalía ni comportamiento extraño del software ha surgido. En caso de que hubiera anomalías, estas deberían ser arregladas en futuras versiones de software y probadas de nuevo [1].

Por último, la evaluación se encarga de que el cliente esté satisfecho con el software que se ha desarrollado. Según los acuerdos a los que se haya llegado con la empresa y habiendo sido testigo de su correcto funcionamiento, da por probado el software, o por fallado, en cuyo caso debería volver a la fase de pruebas y correcciones hasta que cumpla con los criterios y necesidades establecidos por el cliente.

La Figura 5, muestra las fases detalladas que hacen parte del modelo en espiral:

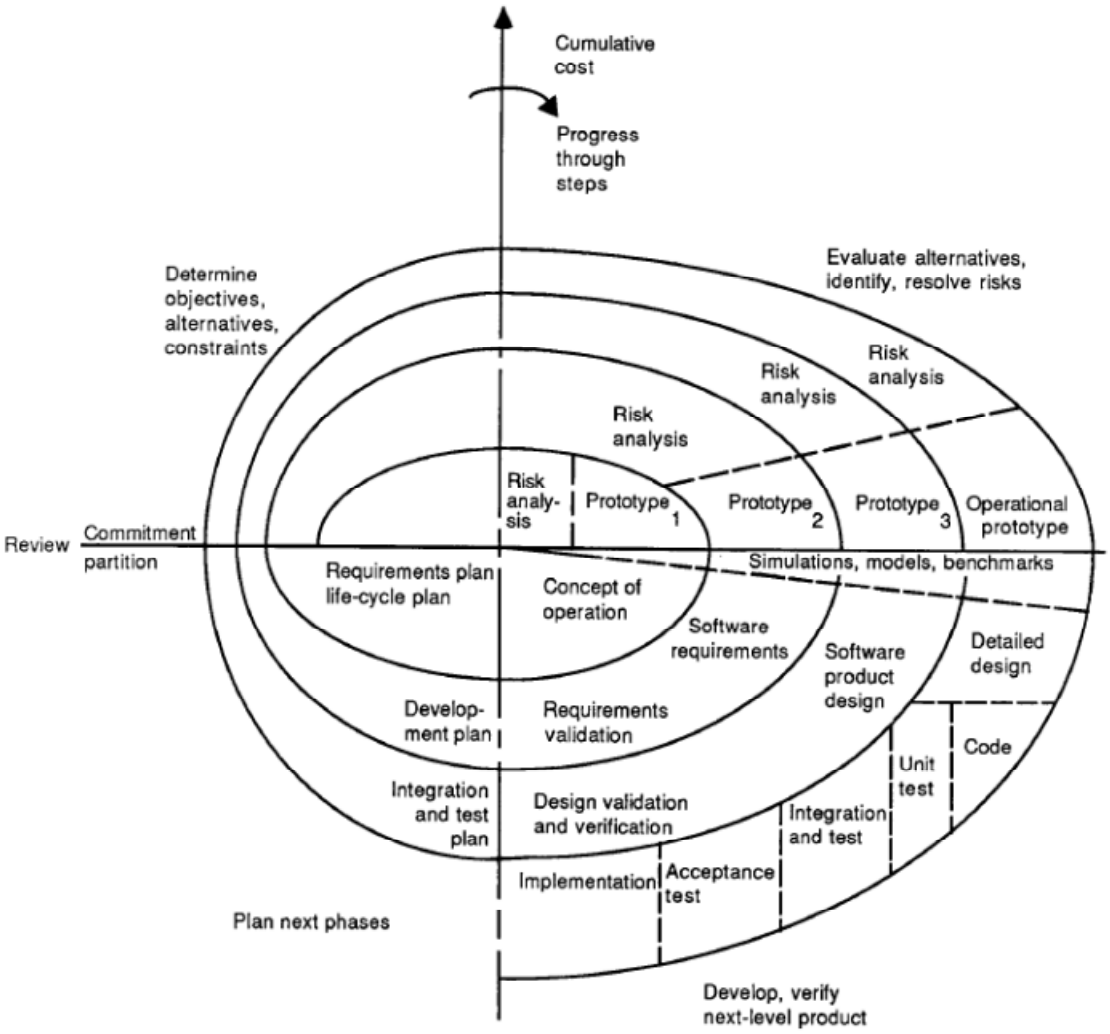

Figura 5: Modelo del ciclo de vida de software en espiral

Ventajas del modelo en espiral:

- Gran profundidad en el análisis de riesgos.
- Muy bueno en proyectos largos.

- Se puede añadir funcionalidad a posteriori de una prueba siguiendo el ciclo de vida en espiral.
- Una temprana versión de software es producida al inicio del ciclo de vida, lo cual hace el proceso más ágil.
- Validación por parte del cliente muy fuerte.

Desventajas:

- Requiere personas con gran experiencia y capacidad en el análisis de riesgos.
- El éxito del proyecto depende en gran medida del análisis de riesgos.
- Pierde efectividad en proyectos pequeños.

2.1.5 Modelo de desarrollo conducido por pruebas (TDD)

Esta metodología se puede definir como "una práctica iterativa de diseño de software orientado a objetos, que fue presentada2 por Kent Beck y Ward Cunningham como parte de Extreme Programming" [2].

El modelo TDD tiene 3 sub-prácticas: automatización de pruebas, Test-first o escritura de las pruebas antes del propio código y refactorización para mantener la calidad del diseño. El modelo de Test driven development (TDD) consiste en la creación de las pruebas que demuestran y validan los requisitos previo al desarrollo del software abstrayendo la información del diseño y contando con el funcionamiento del mismo en el futuro. En este contexto, el software luego se desarrolla en función de la descripción de las pruebas, con lo cual se hace todo de una forma más ágil ya que pueden desarrollarse pruebas y software durante la misma etapa, dando flexibilidad tanto a las pruebas como al software para modificar en tiempo real una parte o la otra para ajustarse y trabajar juntas. De este modo se evitan las nombradas iteraciones de la fase de desarrollo a la de validación y viceversa, las cuales pueden ralentizar un proyecto en gran medida y crear diferentes visiones de un mismo software debido al mal entendimiento de los requisitos o del propio software [1].

Un buen resumen de este modelo sería la frase: "Nunca escribas nueva funcionalidad sin una prueba que falle antes" [2]. En las siguientes figuras (figura 6 y 7) se ilustran un proceso TDD:

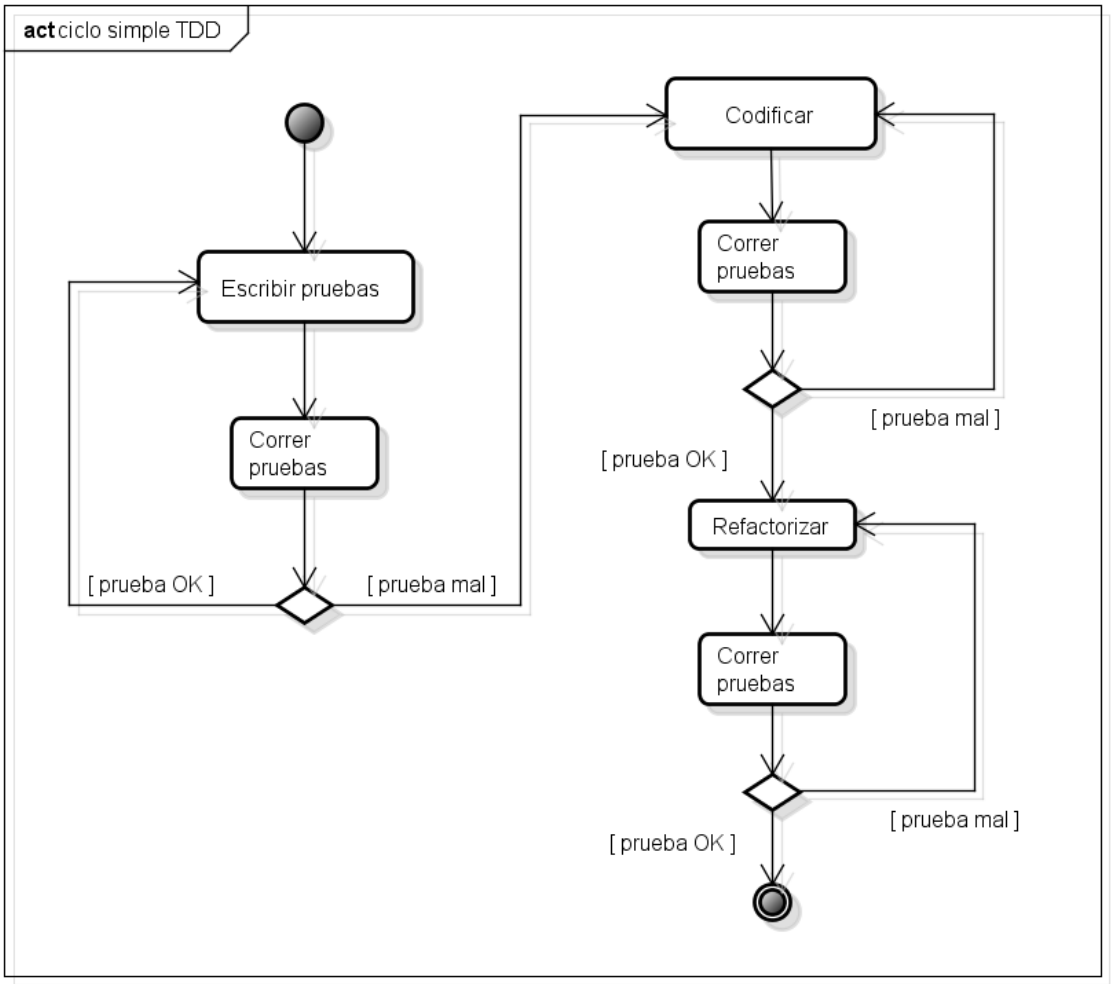

Figura 6: Modelo TDD

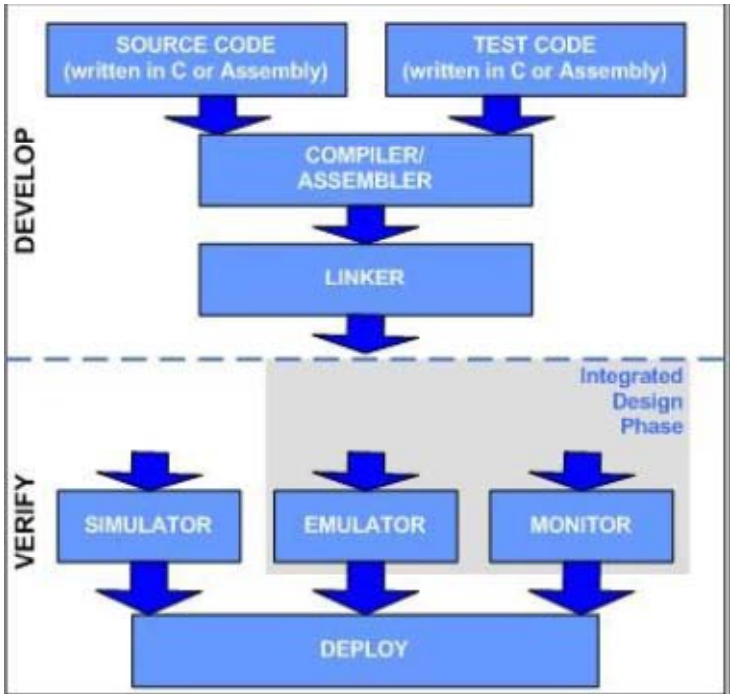

Figura 7: Modelo TDD

Ventajas e inconveniente del modelo TDD:

- Garantiza que las pruebas se ejecutan.
- Diseño enfocado en las necesidades. obliga a que las necesidades reales del cliente sean consideradas primero, obligando a analizar primero qué es lo que realmente se necesita que el código haga y no al contrario.
- Mayor simplicidad en el diseño dado que se enfoca como hemos mencionado en la necesidad del cliente.
- El diseño se va adaptando al entendimiento del problema. Hasta que la prueba no falla no se desarrolla el código lo que produce que el software se adapte poco a poco a las necesidades y no de golpe.
- Mayor productividad. Por lo mencionado anteriormente y producir de forma paralela.

Desventajas:

- Es difícil de implementar en la capa de interfaz de usuario
- La base de datos puede quedar en un estado distinto al que se necesita para hacer la siguiente prueba. Se necesita inicializar de nuevo la base de datos y aumenta la carga de trabajo.
- Se puede perder la visión general. Por ello es recomendable mantener un modelo general.
-

2.1.6 Tipos de Pruebas

Como se puede observar y como es esperable en cualquier metodología de desarrollo software, cada método de desarrollo incluye de una u otra forma fases correspondientes a la Ingeniería de Pruebas. La Ingeniería de Pruebas tiene definido los tipos de pruebas que deben ser realizadas sobre cualquier producto de desarrollo de software. A continuación describimos y explicamos de forma breve cada uno de estos tipos de pruebas;

- **Pruebas unitarias**: Las pruebas unitarias verifican que los subsistemas y componentes software funcionan aislados correctamente: se ejecuta satisfactoriamente la función que se le ha asignado, el flujo de control es correcto dentro del módulo y los datos se calculan con la precisión y en el tiempo requerido [3] [12] [13].

- **Pruebas de regresión:** es la realización de pruebas por las que ya se pasaron para comprobar la estabilidad y madurez del software de cara a una nueva versión.

Sirve para comprobar que los cambios y funcionalidades añadidas del software no han interferido con otras previas. Cuanto menos afecte un nuevo cambio al software más maduro será [12] [13].

- **Pruebas de Integración**: son aquellas que siguen una vez se han realizado todas las pruebas unitarias. Prueban todos los elementos unitarios que componen un proceso, en conjunto y de una sola vez.

  Se usan para verificar que el software en conjunto está funcionando correctamente [12] [13].

- **Pruebas de stress**: estas pruebas se realizan para determinar como de rápido realiza las tareas el sistema bajo condiciones particulares de trabajo, como puede ser a máxima capacidad por ejemplo. También verifica la calidad del sistema como la escalabilidad o la fiabilidad y el uso de recursos. Tienen como objetivo detectar fallos del rendimiento y mejorarlos [12] [13].

- **Pruebas funcionales:** Estas se basan en la revisión de las funcionalidades del software. Para ello se usan modelos de prueba o casos de uso. Son pruebas específicas y exhaustivas para validar el correcto funcionamiento del software y su especificación.

  Para ello se pueden diferenciar varias fases: análisis de requisitos, diseñar el plan de pruebas, Ejecución de las pruebas y manejo de las incidencias y errores ocurridos Estas pruebas podríamos decir, son las genéricas para un software y pueden luego dividirse en las mencionadas anteriormente [12] [13].

3. Estado del Arte

Las herramientas para la gestión de pruebas requieren de un conjunto de actividades indispensables. Dada mi experiencia, he seleccionado algunas de las actividades así como de las características que debe contener cualquier herramienta que se pueda encontrar para el estudio de la ingeniería de Pruebas en entornos académicos:

- **Distribución de la herramienta software:** es importante si la herramienta es de libre distribución o de pago. En particular si tenemos en cuenta que estamos intentando encontrar herramientas que permitan a los ingenieros de prueba junior (a veces con recursos limitados) desarrollar una actividad rigurosa en el área. Por tanto, este criterio supone un factor importante a la hora de elegir una herramienta.

- **Tipo de la herramienta**: una característica importante es el alcance de la herramienta, Cuando hablamos de alcance queremos hacer referencia al tipo de pruebas que la herramienta es capaz de soportar. En particular, es importante determinar si la herramienta se dedica a pruebas funcionales, de carga y rendimiento, a la gestión de las pruebas o si es una herramienta completa que involucra a todas ellas. Adicionalmente, también ha sido incluido como parámetro relevante en este apartado, la funcionalidad de la herramienta para incluir el proceso para la validación del software. Más concretamente, la capacidad de incluir todas las actividades y por tanto, permitir el inicio y fin del proceso de ingeniería de pruebas en un mismo entorno, sin tener que usar diversas herramientas que suponen más dificultad de uso y de aprendizaje. Además de lo que supone contar con los conocimientos limitados que un ingeniero pueda tener de cara al aprendizaje de diversas herramientas, que pueden ser complejas de entender y a la poca experiencia práctica dentro del área.

- **Plataforma disponible:** Como último aspecto relevante, se incluye como criterio el de la disponibilidad de la herramienta para diversos sistemas como pueden ser Linux, Windows, aplicaciones web u otros.

Dados los criterios anteriores, se ha realizado un estudio del estado del arte de estas herramientas en la actualidad. El análisis y estudio de cada una de ellas está basada en los criterios que he definido anteriormente. A continuación se presenta el resultado del estudio realizado.

3.1 Estudio de las herramientas actuales para Ingeniería de pruebas.

En la actualidad, hay muchas herramientas para hacer pruebas, tanto para software libre como para software de pago. A continuación se explicarán algunas de estas herramientas y posteriormente se obtendrá una conclusión sobre lo expuesto en estos puntos. Primero explicaremos las herramientas de libre distribución tomando un subconjunto de ellas, sobre todo las que se han encontrado más interesantes en el contexto de este proyecto. Adicionalmente, se incluirá el análisis de las herramientas con costes asociados.

3.1.1 Herramientas de Libre distribución

Las herramientas de libre distribución o gratuitas son aquellas para las cuales no se necesita pagar una cierta cantidad de dinero para usarlas. Algunas de estas herramientas por ejemplo son: Bugzilla testopia, qaBook, TestLink, Testitool, Selenium, jmeter, Soapui, entre otras. Como se puede observar, existe un número considerable de herramientas pero nos centraremos exclusivamente en aquellas que han parecido tener una funcionalidad más cercana a la buscada por este Proyecto. Por tanto, partiendo de esta selección de herramientas, las dividiremos en grupos según su funcionalidad. Adicionalmente incluiremos un ejemplo de una herramienta en cada subgrupo.

3.1.2 Herramientas de Gestión de pruebas.

En este caso destacan: Bugzilla, qa Manager y qaBook, Squash TM, TestLink y Testitool. En este grupo se sitúan las herramientas usadas para la organización, creación y seguimiento de los de casos de prueba y el establecimiento de su relación con los requisitos. Es decir, básicamente se encargan de crear los procedimientos que seguirán las pruebas, agrupándolas y relacionándolas con los requisitos que se quieren validar.

La que pondremos como ejemplo es TestLink. Además de ofrecer las funcionalidades mencionadas en el párrafo anterior, permite la integración con otros sistemas de seguimiento como Bugzilla o Mantis. Permite agrupar en Planes de prueba y generar informes.

3.1.3 Herramientas para pruebas funcionales.

Para este grupo podemos encontrar: Selenium, Soapui, Watir (Ruby), Solex o SAMIE. Este grupo se encarga de automatizar y ejecutar pruebas creadas. Simplemente se definen unas órdenes (incluidas en los casos de prueba) y la herramienta las ejecuta tal cual como están definidas.

Selenium por ejemplo es una herramienta que permite crear casos de prueba para aplicaciones web. Se divide en dos partes: IDE y Driver. El IDE se encarga de crear los casos de prueba y el Driver, se encarga de ejecutarlos. Cabe la pena destacar que la herramienta permite la automatización de las pruebas para todo tipo de navegadores. Además permite pruebas para aplicaciones móviles con usabilidad de diversos lenguajes como: Python, Ruby, Java, C#, etc.

3.1.4 Herramientas para pruebas de carga y rendimiento.

Algunas de las herramientas encontradas y que pueden agruparse en esta categoría son FunkLoad, loadUI y Jmeter. Estas herramientas permiten hacer pruebas para medir el rendimiento (general o en ciertos estados provocados, como por ejemplo la medición de la capacidad máxima) y la carga soportada dentro de unas condiciones extremas que son especificadas. En el caso específico de Jmeter, ésta herramienta permite realizar pruebas funcionales y de mantenimiento de aplicaciones web (aunque ahora se ha ampliado también a otro tipo de pruebas). Es un entorno basado en Java que trabaja con protocolos HTTP y HTTPS, SOAP, JDBC, JMS, Mail y LDPA.

3.1.5 Herramientas de Pago

Las herramientas de pago son aquellas que pueden adquirirse pagando una licencia para el uso de la misma. A continuación se nombrarán diversas herramientas y sus usos para la Ingeniería de pruebas. En la clasificación de esta categoría nos limitaremos a identificar elementos propios de la herramienta como por ejemplo el ofrecimiento de extras de mantenimiento, soporte, garantía y otros complementos adicionales.

3.1.6 Herramientas de Gestión de pruebas.

En esta categoría, algunas de las herramientas más destacables son: qaComplete, qaBook, SMARTS, practiTest, Zephyr y TestLog.

Herramientas usadas para la organización, creación y seguimiento de los de casos de prueba y el establecimiento de su relación con los requisitos.

3.1.7 Herramientas para pruebas funcionales.

En este apartado de pruebas destacan herramientas software de pago tales como: Rational Robot, Sahi, SoapTest, Test Complete, QA Wizard, Squish o vTest.

Estas herramientas componen la principal manera de ejecutar las pruebas de cara a un software dado, y la manera de acercarse a dicho software adaptándose en cada caso dependiendo de lo que se quiera probar o analizar, así como de los aquellos requisitos que se vayan a validar.

3.1.8 Herramientas para pruebas de carga y rendimiento.

Para finalizar, mencionar algunas herramientas de rendimiento y carga.

- HP LoadRunner
- LoadStorm
- NeoLoad
- WebLOAD Professional
- Forecast
- ANTS – Advanced .NET Testing System
- Webserver Stress Tool
- Load Impact

Estas herramientas destacan por su capacidad de crear ambientes extremos para el hardware y probar de esa manera la capacidad máxima de dicho software. Algunas de estas pruebas son por ejemplo: cantidad máxima de datos transmitidos a máxima capacidad, Comportamiento cuando el sistema está degradado o no está a pleno rendimiento, estabilidad del sistema frente a grandes cargas de datos…

Después de estudiar y analizar el estado del arte de las herramientas para las actividades de la Ingeniería de Pruebas y de acuerdo a la los criterios definidos al inicio de esta sección, se ha diseñado una tabla como guía de comparación de las diferentes herramientas mencionadas anteriormente. Este estudio comparativo incluye los criterios más importantes que he definido como condición necesaria para ser considerados por cualquier herramienta software de entrenamiento para los Ingenieros de Pruebas Junior. A través de este estudio comparativos, hemos contrastado la existencia o no de cada uno de ellos en las herramientas analizadas como es mostrado en la Tabla 1 Un punto adicional y no por ello menos importante, que se ha tenido en cuenta dentro de los criterios en la comparativa que se presenta a continuación es el que una vez se ha iniciado un proceso de pruebas para la validación de un software, la herramienta permita la gestión del ciclo de tareas propias de la Ingeniería de Pruebas.

Tabla 1: Estudio comparativo de las herramientas para Ingeniería de Pruebas

Herramienta	Libre distribución	De pago	Windows	Linux	Web	Otros	Funcional	Carga y rendimiento	Gestión	Completa
Bugzilla	X				X				X	
Qa Manager	X				X				X	
TestLink	X				X				X	
NeoLoad		X	X	X				X		
Jmeter	X		X	X				X		
FunkLoad	X		X	X	X	X		X		
Selenium	X		X				X			
Zephyr		X	X	X					X	
Sahi		X		X			X			
Watir	X		X	X			X			
ANTS		X	X	X	X	X		X		
LoadStorm		X	X					X		
Test Studio		X	X		X					X
TestDroid		X				X	X			

3.2 Resultados del análisis comparativo de las herramientas existentes

De acuerdo a la Tabla 1, la herramienta Test Studio es la herramienta más cercana a la herramienta que consideramos debe servir para el entrenamiento de la Ingeniería de Pruebas. Sin embargo, esta herramienta carece de una gestión de la documentación que en la mayoría de los casos es casi más importante que las propias pruebas en sí mismas. La documentación de la pruebas, son la demostración en forma física de lo bien o lo mal que un software está siendo creado ante el cliente. Es por ello que, en la herramienta que se plantea como una de las contribuciones de este proyecto de fin de frado, se intentará controlar, no solo la ejecución de las pruebas sino además, la facilidad de desarrollar y generar una documentación. Además, la herramienta propuesta debe ser capaz de demostrar los resultados y errores obtenidos para un determinado tipo de software acorde a las pruebas específicas realizadas, así como la cantidad de requisitos demostrados y validados para dicho software.

En este contexto, se puede afirmar que la herramienta a desarrollar en este proyecto es una contribución en el ámbito de las prácticas de asignaturas relacionadas con la Ingeniería de Pruebas, en el sentido que propone solventar en la medida de lo posible, la carencia de otras herramientas para el desarrollo y la ejecución de pruebas (integradas) para el ámbito académico y la iniciación a través de ella y una metodología que la sustenta, en el campo de la ingeniería de pruebas.

4. Metodología Propuesta

4.1 Principios metodológicos

Respecto de los aspectos metodológicos del desarrollo de este proyecto de fin de grado, a continuación se explicarán y fundamentarán las diversas opciones y procedimientos que se han seguido, atendiendo a principios de usabilidad. En este contexto, se ha tomado la decisión de seguir una metodología de desarrollo software interactivo e incremental como la definida en Extreme Programming (XP) siguiendo un enfoque AUP (Agile unified process). Estas opciones se describen en los siguientes sub apartados de una forma más específica.

4.1.1 Ciclo de vida del desarrollo software: iterativo e incremental

Durante todo el proceso de creación se seguirá el modelo iterativo, es decir, cada parte nueva que se cree deberá hacer uso de una anterior creada, de ese modo el resultado final tendrá todos sus componentes enlazados y funcionando dependiendo unos de otros. Además, también se usará el modelo incremental, es decir, el proyecto se irá haciendo más grande cuanto más avance siendo muy pequeño al principio y aumentando de manera progresiva. Así, durante las primeras fases del desarrollo, no se trabajará con grandes cargas de trabajo ni con esquemas realmente complicados. Se busca la sencillez de diseño y la facilidad a la hora de comprenderlo ya que nuestro objetivo es un software educativo, o con fines de aprendizaje para aquellas personas que tengan interés en el área de la Ingeniería de pruebas.

4.1.2 Diseño modular

Para desarrollar el software se ha convenido que la forma más sencilla y la que menos errores y fallas puede provocar de cara a la última fase del proceso de implementación de la herramienta software que es un diseño modular. El diseño modular permite que nuestro proyecto se divida en módulos de tamaño menor que el conjunto del proyecto y que éstos vayan siendo s implementados de manera independiente para que luego sean integrados junto al siguiente a desarrollar formando al final una serie de agrupaciones. Esta opción se ha considerado las más viable debido a que el proceso que acompaña la metodología y que conducirá el desarrollo de la herramienta propuesta, se puede hacer de una forma iterativa (como se menciona anteriormente) y por lo tanto deberá pasar por cada uno de los diferentes módulos creados con anterioridad antes de poder pasar al siguiente. Por ello se puede validar cada módulo antes del desarrollo del siguiente paso de desarrollo, haciendo dicha labor más sencilla y rápida al comprender volúmenes de código menores.

Con este desarrollo, se pretende conseguir una fluidez y sencillez mayores que desarrollando la herramienta como un conjunto y además, tener una visión más específica y detallada de cada módulo de la herramienta por separado. De esta manera, la tarea de depurar y limpiar el código de la herramienta software se realizará de una forma que desde mi punto de vista y mi experiencia profesional considero más adecuada.

4.2 Fases de la Ingeniería de Pruebas: metodología propuesta

Tomando en consideración todos los conocimientos adquiridos del sistema de funcionamiento de la ingeniería de pruebas en un ambiente de trabajo real, y simplificándolo de manera que sea sencilla de comprender, aplicar y desarrollar en un ambiente académico y por tanto didáctico, el modelo y metodología propuestos se explicarán a continuación de una forma detallada.

Se ha considerado el proceso de la ingeniería de pruebas como algo en constante desarrollo, por ello la mejor forma de representarlo será con un modelo cíclico.

Las fases a desarrollar en el próximo apartado sobre nuestro modelo son:

- La creación de requisitos,
- la creación de las pruebas,
- la validación de las pruebas y requisitos, l
- a creación de anomalías en caso necesario y el cierre de las mismas mediante diferentes versiones del software,
- La generación y registro de los resultados obtenidos para cada prueba,
- La visualización de dichos resultados frente al cliente con carácter formal,
- La documentación asociada a cada prueba
- El registro del resultado y de las anomalías obtenidas durante el proceso y
- La validación de cada documento con el grupo de calidad.

A partir de este punto, el ciclo volvería a comenzar, tras el cual, una vez realizada una vuelta y haber descubierto los fallos del software una nueva versión de software y requisitos podría ser necesaria para solventar estos descubrimientos.

A continuación se detallará cada fase por separado más específicamente. Es notable reseñar, que a veces desde la fase de generación de anomalías, una nueva rama del ciclo puede surgir. En ella, si no hay cambios de requisitos necesarios, el equipo de software puede trabajar de forma independiente corrigiendo los fallos surgidos dando así lugar a una nueva versión de software que necesitara volver a ser validada volviendo a la fase de validación de pruebas (en este caso las que fallaran por una anomalía de tipo software).

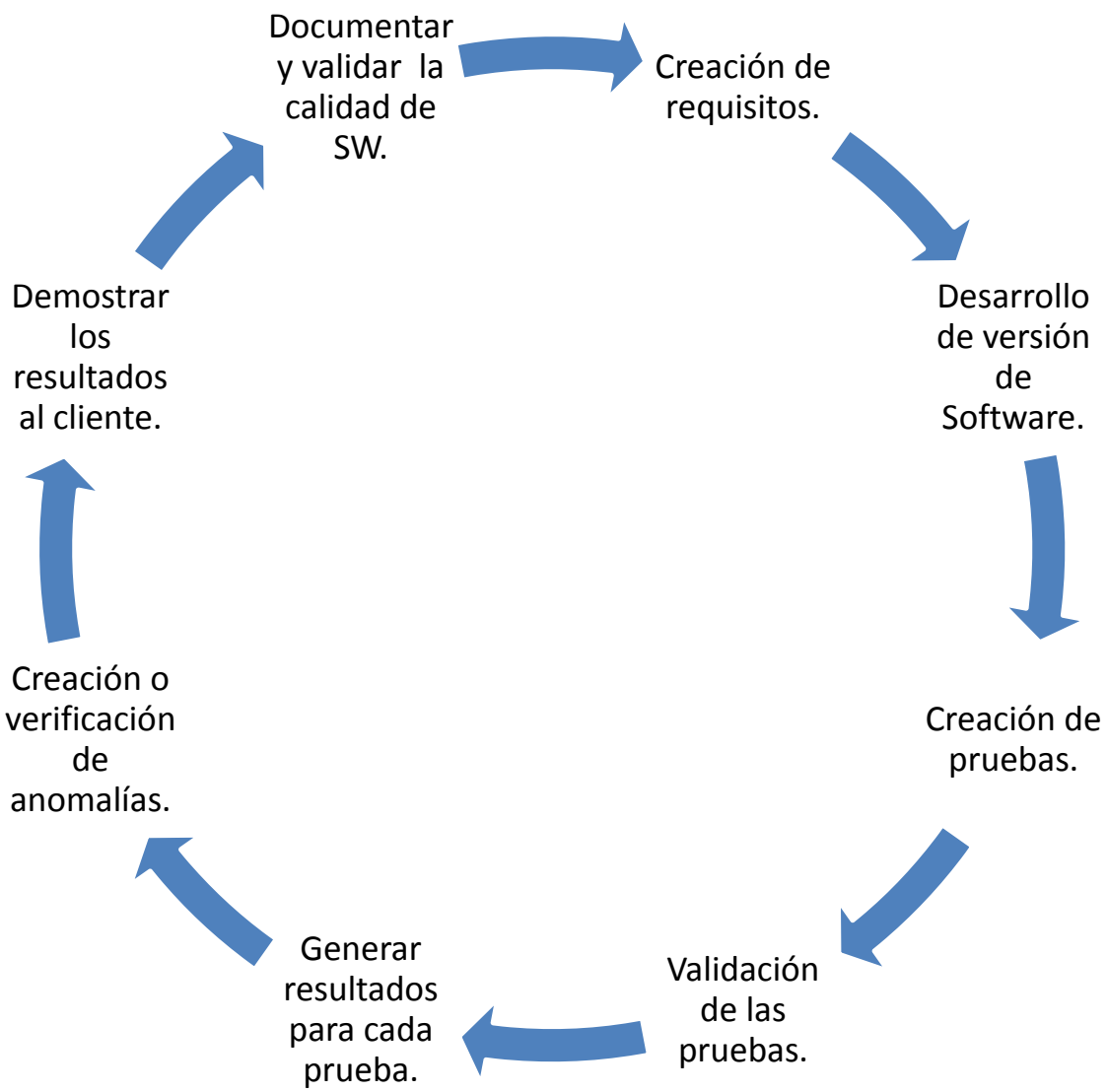

Figura 8: Metodología propuesta.

4.3 Explicación de la metodología propuesta:

Fase inicial:

I. Creación (o modificación) de requisitos

Esta fase podría considerarse la inicial y la que implica un comienzo de cada ciclo del método, aunque no siempre tiene por qué ser así. En este momento es donde cliente, el equipo de especificación de requisitos y los Project manager necesarios, se juntarían y concretarían cómo debe funcionar el software. De modo que se especificarán todos los requisitos necesarios. Esto deberá realizarse de forma atómica, para que sean entendibles y puedan llevarse y plasmarse en una versión de software. Esta es una de las fases más

importantes de todo el método y por ello se ha considerado como la inicial de cada ciclo. Una mala gestión, creación e interpretación de los intereses del cliente (ya que a grandes rasgos es una plasmación de las peticiones del cliente) llevará a la generación de un mal software y por tanto, a un aumento innecesario de ciclos. Esto genera en el proyecto un aumento en versiones de software baldías para corregir la mala o incorrecta gestión de los requisitos y anomalías que no satisfacen las verdaderas necesidades del cliente. A su vez, un mal entendimiento de los requisitos por parte del equipo encargado de la creación del software llevará a un mayor número de versiones ya que se necesitarán varios ciclos para arreglar el mal entendimiento de los mismos.

En particular, durante esta fase, se ha de poner un gran empeño en que los requisitos queden claros y entendidos por todas las partes del proyecto (cliente, Project managers, equipo de especificación y equipo de desarrollo) para ahorrar tiempo, problemas y posiblemente dinero en un entorno de trabajo real en el futuro. Es importante que los Project manager estén al tanto de la creación y modificación de requisitos ya que es el momento de discutir con el cliente una versión realista del sistema que coincida con las expectativas del cliente.

Como puntos importantes a ser resaltados en esta fase, se puede afirmar que un proyecto se podría medir lo bien o mal especificado que está dependiendo de varios puntos los cuales los más comunes serían: *número de versiones de los requisitos, es decir, el número de veces que los requisitos han tenido que modificarse o ha sido necesario crear o borrar requisitos para el apropiado funcionamiento del software y la validación de resultados y el proyecto y la cantidad de versiones del software*. Aunque este último punto no depende solo de lo “bien” o “mal” que esté especificado el producto, influye en la cantidad de versiones de software que deberán realizarse ya que por cada modificación de requisito el software podría pasar a comportarse de una manera diferente y por tanto, será necesaria la creación de una nueva versión de software.

II. Creación de una versión de Software:

Durante esta fase del proyecto el equipo de desarrollo, durante el primer ciclo de la metodología propuesta, leerá y entenderá los requisitos especificados y una vez entendidos los llevará a una versión software de los mismos. Dentro del equipo de desarrollo se harán unas pruebas internas mínimas para comprobar por encima el funcionamiento básico del mismo pero no serán pruebas de mucha profundidad ya que esa es la tarea del ingeniero de pruebas.

En caso de no ser el primer ciclo de la metodología, lo que tendrá que resolver el equipo de desarrollo son todos los fallos que hayan podido ocasionarse durante la validación de la primera (o posteriores) versiones de software así como la posible inclusión o modificación de los requisitos existentes. Lo anterior podría entonces generar nuevas funcionalidades

necesarias en el software o modificar las ya existentes. También puede darse el caso de que en vez de añadir una funcionalidad ésta se elimine, en cuyo caso sería el proceso de eliminar del software dicha funcionalidad. Otra tarea del equipo de desarrollo durante esta fase, puede ser el dejar lista la versión de software para una release (liberación de la versión) formal. Entonces, podría generarse una primera versión ligera del software, no completa pero si funcional, para que se vaya probando, y de esta manera seguir trabajando en paralelo generando nuevas versiones de software mientras se van realizando las pruebas hasta completarlo o liberar dicha versión.

De este modo el ciclo se desviaría momentáneamente, a saber: ya no se realizaría una misma actividad sino dos de forma simultánea enlazando esta tarea y la posterior en un único salto de la metodología. Por un lado, el equipo de pruebas seguiría creando y validando pruebas y requisitos. Por el otro lado, el equipo de software seguirá avanzando en la resolución de fallos y el añadido o borrado de funcionalidades simultáneamente. Esto permite seguir trabajando a ambos equipos muy rápido, en caso de que por ejemplo se vaya apurado con las fechas de entrega. No obstante, puede ocasionar conflictos entre lo que se quiere probar y lo que aún no está implementado, se está implementado o se ha borrado de una versión a otra y no se avisa al equipo de pruebas. Es decir, por un lado es una forma de trabajar mucho más ágil y rápida pero se necesita tener mucho más control en lo que hace cada equipo, ya que sino ambos equipos podrían tener conflictos. Por tanto, es importan definir un protocolo de manejo de versiones que evite estos conflictos.

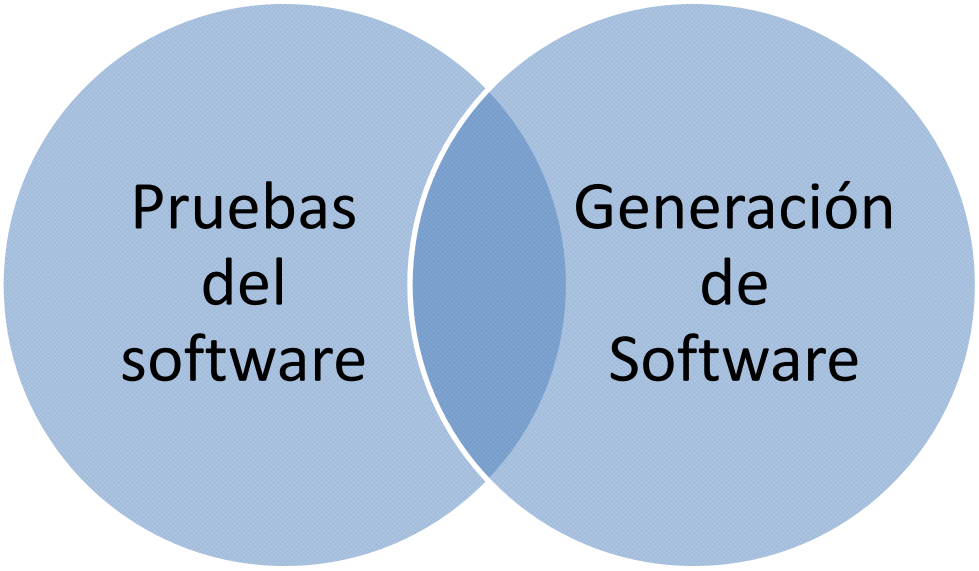

Figura 9: intersección entre pruebas y software

III. Creación de las pruebas:

En esta fase pueden surgir diferentes grupos de pruebas dependiendo del software a probar y del enfoque que se le quiera dar. En este caso, como nuestra metodología se asimilaría a un TDD (creación de pruebas pensando en el funcionamiento del software) esta fase podría ser simultánea a la creación del software o incluso anteponerse o posponerse hasta tener una primera versión de software para crear pruebas más detalladas, precisas y válidas.

Como se ha explicado en el apartado 2.1.6 de este documento, las pruebas pueden ser de diversa índole: unitarias, de regresión, funcionales, de Integración, etc. Cada una de ellas tiene un rol específico dentro de la Ingeniería de Pruebas y por tanto deberían separarse de una forma adecuada por grupos. De esta forma, su correspondiente gestión (creación, ejecución y validación) puede realizarse de una forma organizada y ordenada. Este es un factor clave dentro de la metodología que proponemos dado que puede darse el caso que la gestión de las pruebas lleven un ritmo muy lento y sinsentido a la hora de ejecutarlas. En muchos casos, esto produce que se tengan que saltar unas para poder realizar las otras. La validación de las pruebas debería ser una fase fluida, que no se haga pesada y que resulte clara de tal forma que posteriormente, pueda ser registrada y explicada formalmente frente al cliente. Si durante esta fase las pruebas no fueran precisas y claras dará una sensación al cliente de que el software menos maduro. Hay que tener en cuenta que el cliente no tiene por qué ser un experto en el software desarrollado y por tanto se deberán crear pruebas que cualquiera pudiera ejecutar, llegando a un nivel tal que se pueda describir de forma detallada, paso por paso, cada entrada de datos y cada acción que se lleve a cabo.

Durante la creación de dichas pruebas es importante tener en cuenta el funcionamiento del software ya que al fin y al cabo, cada prueba validará una función del software. Dicha función estará especificada en algún un requisito, de modo que al final una prueba debería siempre estar validando al menos un requisito específico, aunque pueden ser varios a la vez.

La herramienta que se propone en este Proyecto de Fin de Grado, permite que en el momento de creación de la prueba, se le pueda asociar a ésta uno o más requisitos los cuales están almacenados en la base de datos de la herramienta. De esta forma, cuando se ejecute la prueba, se podrá ver claramente el requisito que está validando.

En resumen y como conclusión de este punto; la creación de las pruebas debe estar siempre en concordancia con el funcionamiento del software y la especificación de los requisitos. Estos tres pilares forman un triángulo necesario para conseguir que un proyecto salga adelante con criterios de calidad, ya que si cada uno hace, especifica, entiende o desarrolla su parte de una forma independiente a las otras dos se creará una separación entre las distintas fases de la metodología y el proyecto a la larga costará más

tiempo y dinero. Una parte importante de la especificación de la metodología y que hace que esta pueda ser útil o pueda ser aplicada conforme a la filosofía con la cual ha sido planteada, es la concienciación de los usuarios de ella. Si desde un inicio todas las partes deben estar concienciadas y sus procesos de comunicación son efectivos y eficientes, una metodología aplicada de forma rigurosa y sistemática puede dar lugar a los resultados esperados. Además y específicamente para esta fase, las pruebas deben ser claras, rápidas, sencillas y precisas. Deben poder ejecutarse en su mayoría una tras otra de una forma eficiente, y deben estar organizadas de modo que estén relacionadas con un grupo determinado (como se ha explicado al inicio) para no crear una sensación de descontrol y caos cando estás se demuestren ante el cliente.

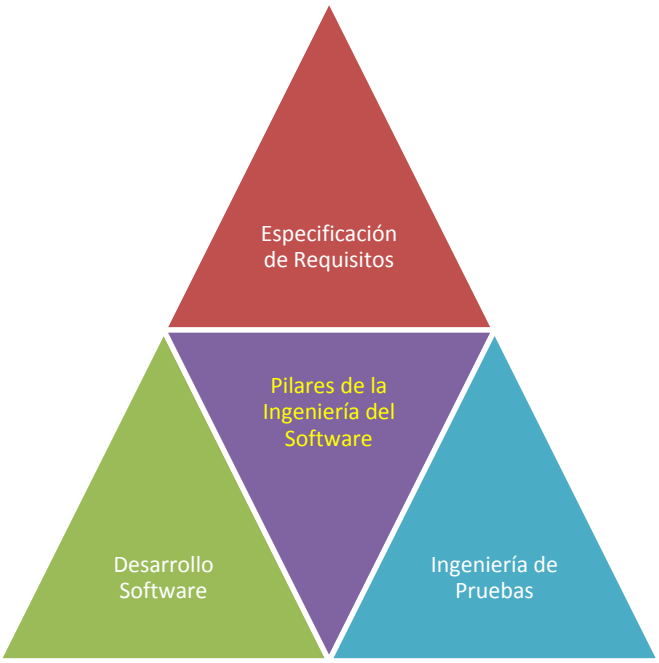

Figura 10: Pilares fundamentales de la Ingeniería Software.

IV. Validación de las pruebas:

En esta fase de la metodología, es cuando se unen las fases de desarrollo de software y creación de las pruebas. Con el software proporcionado por el equipo de desarrollo, los ingenieros de pruebas deben de validar los requisitos asociados con cada prueba previamente definida usando el software proporcionado. Esta fase como se ha explicado anteriormente, puede ir de forma simultánea mientras se crean pruebas y se genera nuevo software dando lugar a una fase dividida en tres partes como ilustra la Figura 9. En esta Figura se puede observar que cada equipo por separado seguirá trabajando en su tarea específica, pero a la vez hay un punto donde se juntan y debe

colaborar el uno con el otro para la creación de las pruebas, su correcta validación y ejecución. Adicionalmente, en conjunto, estos grupos deben detectar los posibles problemas y anomalías que pudiera mostrar el software generado. Sin esta fase no puede darse lugar la siguiente del método propuesto.

V. Generar resultados para cada prueba:

Este proceso es uno de las más simples de la metodología y puede ocurrir simultáneamente con la ejecución de las pruebas para un mayor control, aunque en la Figura 8 se haya separado para explicarlo con mayor profundidad. Simplemente, consiste en llevar actualizados los resultados de cada prueba dependiendo de si durante su ejecución la prueba fue satisfactoria o si hubo algún fallo. En caso de ser un resultado positivo, la prueba se considerara "pasada" y simplemente habrá que volver a probarla frente al cliente de manera formal. Si fue un resultado negativo, se dejará constancia de ello con un resultado fallado y trazando además una anomalía al paso como se explica para la siguiente fase de la metodología propuesta.

VI. Creación o verificación de anomalías:

Para explicar esta fase es conveniente separarla en dos partes a saber: la creación y la verificación de anomalías.

Partiendo de la fase anterior, donde cada prueba se valida con la versión de software correspondiente, puede ocurrir que se detecte un fallo o anomalía. Estas anomalías pueden ser trazadas a prueba o no. Las ***trazadas a prueba*** son aquellas que hacen que la prueba en si misma falle y se le asigne un resultado negativo. El resto de anomalías se han encontrado durante la ejecución de una prueba pero no tienen que ver necesariamente con la misma. Por ejemplo: si la prueba consiste en hacer un login, pero al realizar el login se comprueba que desaparece el título de alguna función o botón, *esto sería una anomalía no trazada a prueba*. La diferencia con las que están trazadas a prueba es que tienen una importancia menor, por así decirlo, son más leves ya que éstas necesitarán también ser corregidas pero no necesariamente mostradas al cliente.

El proceso a seguir para gestionar esta fase es el siguiente:

- Se asociaría a la prueba un resultado fallado, se abriría una nueva anomalía y se esperaría a la corrección de la misma. Antes se debe asegurare consultando con

los otros equipos (desarrollo y especificación) que de verdad es una anomalía, ya que puede darse el caso que la prueba este mal diseñada acorde al software. Una anomalía no es nada más que la documentación del problema en sí mismo y la forma en la que se ha detectado, es decir, el procedimiento de la prueba. Con esta información el equipo de desarrollo lo tendrá en cuenta para la generación de la siguiente versión de software, en la cual pueden incluirse diversas anomalías ya corregidas. Esto nos lleva al siguiente punto.

- Durante la verificación de las anomalías, se cogerán todas las anomalías resueltas y se volverán a probar con su correspondiente procedimiento, es decir, con su prueba asociada. Si la validación tiene un resultado satisfactorio, entonces la prueba puede darse como pasada y la anomalía como corregida. En caso de que siga fallando se volverá a poner el estado de la anomalía como fallando. Este proceso es uno de los más importantes ya que cada anomalía resuelta le aportara madurez y consistencia al software. Cuanto menos anomalías por versión del software ocurran, más maduro será el software de cara al cliente.

VII. Demostrar los resultados al cliente:

Este proceso es uno de los más importantes del proyecto, junto con la buena especificación de requisitos, ya que en ambos influye el cliente del proyecto y tienen un carácter más formal. Durante esta prueba, se validará cada prueba creada frente al cliente para demostrar que realmente son correctas. Pueden ocurrir distintos resultados:

1. **Las pruebas que tiene como resultado la etiqueta “pasada“vuelven a ser satisfactorios**. En este caso se dejará constancia (en algún material adicional o en una versión más avanzada de la herramienta, ya que actualmente no se ha considerado el carácter formal o no de los resultados) del nuevo resultado, esta vez con carácter formal. Esto significa que este resultado ahora sí será válido, y no será necesario volver a validarlo excepto en caso de hacer pruebas de regresión.

2. **Una prueba con resultado “pasado” falla**. En este caso se siguen los mismos pasos que si fallara de forma interna pero la diferencia sería que al igual que un resultado tiene carácter formal, la anomalía descubierta pasaría a tener una mayor importancia, ya que ha sido descubierta durante una prueba formal.

3. **Un resultado “fallado” aún no se ha arreglado**. Existen dos líneas de acción para estos casos: la primera sería acordar con el cliente saltar la ejecución del paso ya que aún no está arreglado y va a seguir fallando o probarlo, para que vea donde está el fallo. En estos casos dependen varios factores para la elección, ya sean, lo que se acuerda por contrato, el tiempo que lleve la

ejecución de dichos pasos y la formalidad que se le quiera dar a las anomalías. Si se ha acordado con el cliente saltar las pruebas falladas de antemano, no hay ninguna duda. Si se prevé una falta de tiempo al ejecutar las pruebas falladas, entonces podría decidirse saltarse las falladas sin corregir. En el tercer caso, si un resultado falla frente al cliente, habría que cambiar el carácter de la anomalía por uno más formal y esto no siempre es conveniente. A mayor cantidad de anomalías "formales" más prioridad le tienes que dar a las mismas y por tanto restas tiempo a otras actividades, además si hay muchas anomalías con este carácter el cliente puede llevarse una mala impresión del software.

4. **Un resultado fallado ha sido arreglado**, es decir, la anomalía trazada a la prueba ha sido corregida. En este caso se validará dicha prueba frente al cliente y procederá acorde al resultado. Si es pasada se procederá como en el punto 1, si es fallada como en el punto 2.

VIII. Documentar y validar la calidad del Software y de las pruebas:

Esta es la última fase y es un poco el resumen de un ciclo de la metodología (entendemos por ciclo el haber pasado una vez por cada fase). Tiene un carácter formal y sirve para plasmar todo lo realizado durante un ciclo. En esta fase podemos incluir la documentación de los resultados obtenidos frente al cliente, cantidad de anomalías abiertas aún, cantidad de anomalías corregidas en el ciclo, pruebas aún pendientes de ejecutar (falladas o no probadas por falta de tiempo), pruebas no necesarias de una nueva ejecución (a menos que se decida hacer una regresión), etc. Esta fase se puede considerar como la fase que gestiona la documentación. Tiene que quedar constancia a poder ser por escrito, de cómo ha ido el proyecto hasta la fecha, considerando todos los aspectos que se consideren necesarios o hayan sido pactados con el cliente. En esta fase suelen salir una serie de indicadores y porcentajes relevantes para comprobar la calidad del proyecto como pueden ser: ratio de resultados pasados, resultado de fallados, cantidad de anomalías, requisitos sin probar, requisitos sin implementar, entre otros, los cuales son resultados de carácter más estadísticos y quizá más visibles para el cliente.

Resumiendo, es una fase de formalización global del proyecto mediante documentos y valores estadísticos para tener evidencia de lo "bien" o "mal" que va un proyecto.

4.4 Conclusiones de la metodología propuesta:

1. Aunque en la Figura 8 se ha presentado la metodología como de forma cíclica, ha sido meramente por simplicidad visual y por el carácter recursivo que tiene dicha metodología. Como se ha explicado anteriormente una o más fases

pueden agruparse y realizarse de manera simultánea generando cambios en la presentación del modelo.

2. Toda actividad de carácter formal, es decir, toda aquella en la que influya el cliente debe dársele un carácter y una rigurosidad mayor que al resto de fases, ya que en ellas estás demostrando la seriedad de los equipos y la madurez del software.

3. Todos los equipos y personas involucrados en el proyecto deben estar en constante comunicación y deben trabajar juntos para mejorar la calidad del producto.

4. La herramienta aportada por este proyecto de forma complementaria a la metodología propuesta cubre la mayoría de funcionalidades propuestas pero no todas las especificadas. Es por ello que se pueden encontrar diferencias a la hora de usarla junto con la metodología como pueden ser: la formalidad de resultados y anomalías o las transiciones posibles de una anomalía (abierta, corregida, validada, devuelta para corregir, etc.).

5. Todas estas fases han sido creadas simplificando las ideas obtenidas de la experiencia laboral en un entorno real de trabajo para proyectos grandes, y por lo tanto puede variar para entornos o proyectos pequeños. De hecho en entornos pequeños debería simplificarse ya que no habrá diversos equipos, pero sí diversas personas trabajando. Para estos casos pueden repartirse los roles y tareas de una forma semejante a la metodología propuesta: Cliente, Project manager, especificación de requisitos, desarrollo de software e Ingeniero de pruebas. De hecho, la herramienta está pensada de manera simplificada y debería ser de mayor uso en proyectos pequeños, ya que en los proyectos grandes está herramienta no tiene tanta profundidad en el área al estar pensada para un ámbito universitario, estudiantil y de aprendizaje. Cubre lo necesario para seguir la metodología propuesta en proyectos pequeños y para demostrar a los alumnos o personas interesadas en la materia de la Ingeniería de pruebas lo que pueden encontrarse en el mundo laboral a pequeña escala.

5. Análisis: Ingeniería de requisitos

5.1 Educción de requisitos.

Se desea "diseñar e implementar la herramienta software NOVATests, la cual es planteada como un software con carácter de repositorio de pruebas con objetivo de uso de estudiantes de ingeniería o Ingenieros recién licenciados (ingenieros prueba junior)". Para ello, debe tener asociada una base de datos y se compondrá de cinco funciones o menús, a saber:

- Tests,
- Resultados,
- Requisitos,
- Incidencias y
- Salir de aplicación.

En el apartado de ***Tests*** se podrán realizar las siguientes funciones:

- Crear prueba,
- modificar prueba,
- borrar prueba,
- añadir requisito,
- añadir incidencia,
- añadir resultado,
- ver prueba y
- cerrar ventana.

La ventana mostrará las pruebas existentes con su identificador, la entrada y salida de datos que se realizan en la prueba y los requisitos o anomalías asociadas.

Resultados tendrá las funciones:

- Añadir resultado,
- Modificar resultado,
- Borrar resultado y
- cerrar ventana.

A su vez esta ventana mostrara las pruebas existentes, la prueba seleccionada y el resultado de la prueba seleccionada.

Requisitos tendrá las funciones:

- Nuevo requisito,
- Ver requisito,
- Modificar requisito,
- Borrar requisito y

- Cerrar ventana.

Esta ventana mostrara los requisitos existentes en la base de datos y el contenido del requisito seleccionado.

Por último la ventana ***Incidencias*** podrá usar las funciones:

- Añadir Incidencia,
- Ver incidencia,
- Modificar incidencia,
- Borrar incidencia y
- Cerrar ventana.

Y a su vez esta ventana mostrará las incidencias existentes, su descripción y si tienen alguna prueba asociada.

Generalidades: cada vez que se desee salir del programa o realizar cambios o borrar algún elemento deberá mostrarse una ventana de confirmación. A su vez no se deberá poder modificar elementos no seleccionados ni borrarlos. El idioma del programa será en inglés para comprensión global. La base de datos se deberá poder arrancar a parte y podrá seleccionarse su ubicación. [4][5].

5.2 Especificación de requisitos del entorno software.

5.2.1 Requisitos no funcionales:

Requisito 1: Deberá poder ejecutarse el software NOVATests en Sistema operativo Windows.

Requisito 2: Deberá abrirse el software desde un archivo ejecutable (.jar)

Requisito 3: La base de datos será ajena al programa y deberá iniciarse por separado del software.

Requisito 4: La base de datos se desarrollara mediante MongoDB y deberá tener un archivo de arranque (.bat) en el cual se especificará la ruta de la base de datos (/data por defecto).

Requisito 5: El software de NOVATests se desarrollara en lenguaje JAVA usando el programa NetBeans 8.0.1.

Requisito 6: El programa deberá ser estable y solo se cerrará cuando el operador seleccione la función “Salir de la aplicación” o se cierre con el botón CERRAR de la ventana correspondiente.

Requisito 7: El programa deberá funcionar fluido sin pérdidas de funcionabilidad para volúmenes de datos relativamente grandes.

Requisito 8: Cualquiera con acceso al programa será capaz de abrirlo y ejecutarlo así como el acceso a inicializar la base de datos.

Requisito 9: No será necesario el uso de LOGIN y PASSWORD para el acceso al programa.

Requisito 10: El programa deberá permitir futuras modificaciones para añadir/eliminar funcionalidad.

Requisito 11: El programa deberá mostrar mensajes de error o advertencia cuando sea necesario.

Requisito 12: Cada ventana nueva que se muestre deberá aparecer centrada en la pantalla y además, deberá bloquear el acceso a una ventana abierta previamente mostrada en background.

5.2.2 Requisitos Funcionales:

Relativos al menú Inicial:

Requisito 13: El programa, recién arrancado, deberá mostrar la "Ventana Inicial" con un mensaje de bienvenida. Dicho mensaje contendrá un resumen breve de la aplicación y su uso.

Requisito 14: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana, mediante la función de la barra de menús, mediante el atajo de teclas CTRL+E o mediante el botón [Cerrar aplicación].

Requisito 15: En el momento que se desee cerrar la "Ventana Inicial", se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar el cierre de la aplicación de la siguiente forma:

- **Sí:** Se cerrará la aplicación, sin perder los datos de la base de datos inicializada.
- **No:** Se volverá a la ventana inicial.

Requisito 16: La "Ventana Inicial" Deberá darte la posibilidad de continuar con la base de datos cargada, según el estado que estuviera o de inicializar de nuevo la base de datos actual, borrando su contenido y partiendo de 0.

Requisito 17: Pulsando el botón [Continuar (vieja DB)] o seleccionando la función en la barra de menús o mediante el atajo de teclado CTRL+L, se cerrará la ventana actual y se abrirá la ventana "Menú Principal".

Requisito 18: Pulsando el botón [Nueva Base de Datos] o seleccionando la función en la barra de menús o mediante el atajo de teclado CTRL+N, la base de datos actual será

inicializada de cero, es decir, se borrarán todos los datos actuales y quedará una base de datos en blanco.

Requisito 19: Una vez se pulse el botón mencionado anteriormente (Nueva Base de Datos) se deberá mostrar de inmediato y de forma previa a los cambios un mensaje de confirmación dando la posibilidad de aceptar o rechazar está funcionalidad de la siguiente manera:

- **Sí:** Los datos de la base de datos se borrarán y se abrirá de forma inmediata la ventana de "Menú Principal".
- **No:** Se volverá a la ventana anterior sin realizar ningún cambio.

Requisito 20: Si por cualquier razón se abriera la ventana "Menú Principal", no habría forma de volver a abrir "Ventana Inicial" si no es cerrando y volviendo a abrir la aplicación de nuevo. Del mismo modo, dicha ventana no se mostrará nunca más durante la ejecución del programa ni siquiera en plano de fondo.

Relativos al Menú Principal:

Requisito 21: Tras haber seleccionado una opción de la Ventana Inicial exceptuando la opción de cerrar aplicación, deberá mostrarse la ventana "Menú Principal".

Requisito 22: deberá mostrarse la ventana "Menú Principal" con un mensaje de bienvenida. Dicho mensaje contendrá un resumen breve de la aplicación y su uso.

Requisito 23: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cerrar aplicación].

Requisito 24: En el momento que se desee cerrar la ventana "Menú Principal", se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar el cierre de la aplicación de la siguiente forma:

- **Sí:** Se cerrará la aplicación, sin perder los datos de la base de datos inicializada.
- **No:** Se volverá a la ventana "Menú Principal".

Requisito 25: Pulsando el botón [Requisitos] Se abrirá la ventana "Requisitos" donde se podrá modificar, crear y borrar los requisitos de la Base de datos.

Requisito 26: Pulsando el botón [Pruebas] Se abrirá la ventana "Pruebas" donde se podrá modificar, crear y borrar las pruebas de la Base de datos así como trazar resultados, requisitos e incidencias a las mismas.

Requisitos 27: Pulsando el botón [Resultados] Se abrirá la ventana "Resultados" donde se podrá modificar, crear y borrar los resultados de las pruebas de la Base de datos.

Requisitos 28: Pulsando el botón [Incidencias] Se abrirá la ventana "Incidencias" donde se podrá modificar, crear y borrar las incidencias de la Base de datos.

Relativos a la ventana de Requisitos:

Requisito 29: Tras haber seleccionado la opción de Requisitos en el menú principal deberá mostrarse la ventana "Requisitos".

Requisito 30: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cerrar ventana].

Requisitos 31: Al abrirse esta ventana mostrara una lista de todos los requisitos existentes y un cuadro de texto disponible para mostrar la descripción de los requisitos.

Requisito 32: La ventana "Requisitos" dispondrá de 5 botones disponibles: [Ver Requisito], [Nuevo requisito], [Modificar Requisito], [Borrar Requisito] y [Cerrar Ventana]. Dichos botones se comportaran de la siguiente manera:

- **Ver Requisito:** Se mostrará la descripción del requisito seleccionado en el correspondiente cuadro de texto.
    - Nota: En caso de que no se haya seleccionado ningún requisito aparecerá un mensaje de aviso mostrando: "Ningún requisito se ha seleccionado para mostrar".
- **Nuevo Requisito:** Se abrirá la ventana "Nuevo Requisito" permitiendo la posibilidad de crear un nuevo requisito.
- Nota: Ver requisitos correspondientes a la ventana "Nuevo requisito".
- **Modificar Requisito:** Se abrirá la ventana "Modificar Requisito" permitiendo la posibilidad de modificar un requisito existente.
    - Nota 1: En caso de que no se haya seleccionado ningún requisito aparecerá un mensaje de aviso mostrando: "Ningún requisito se ha seleccionado para modificar".
    - Nota 2: Ver requisitos correspondientes a la ventana "Modificar requisito".
- **Borrar Requisito:** permite la posibilidad de Borrar un requisito existente.
    - Nota 1: En caso de que no se haya seleccionado ningún requisito aparecerá un mensaje de aviso mostrando: "Ningún requisito se ha seleccionado para Borrar".
    - Nota 2: Ver requisito asociado -Requisito 5-.
- **Cerrar Ventana:** Se cerrará la ventana actual volviendo a la ventana "Menú principal".

Requisito 33: En el momento que se desee Borrar un requisito (pulsando el botón [Borrar requisito]), se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar borrado del requisito seleccionado de la siguiente forma:

- **Sí:** Se borrará el requisito, eliminándolo de la base de datos.
- **No:** Se volverá a la ventana "Requisitos" sin realizar ningún cambio.

Requisito 34: Al pulsar doble click sobre la ID de un requisito mostrado en la lista izquierda de la ventana, está función actuará como si pulsáramos el botón [Ver Requisito]. De este modo su descripción se mostrará en el cuadro de texto correspondiente.

Relativos a la ventana de Resultados:

Requisito 35: Tras haber seleccionado la opción de Resultados en el menú principal deberá mostrarse la ventana "Resultados".

Requisito 36: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cerrar ventana].

Requisito 37: Al abrirse esta ventana mostrara una lista con todas las pruebas existentes en la base de datos así como dos cuadros de texto no editables donde se mostrara la ID de la prueba y su resultado.

Requisito 38: La ventana "Resultados" dispondrá de 5 botones disponibles: [Ver Resultado], [Añadir Resultado], [Modificar Resultado], [Borrar Resultado] y [Cerrar Ventana]. Dichos botones se comportaran de la siguiente manera:

- **Ver Resultado:** Se mostrará la ID de la prueba seleccionada en el correspondiente cuadro de texto y su resultado actual en el cuadro de texto colindante (en blanco en caso de no tener resultado).
- Nota: En caso de que no se haya seleccionado ninguna Prueba aparecerá un mensaje de aviso mostrando: "Ninguna Prueba se ha seleccionado para mostrar".
- **Añadir Resultado:** Se abrirá la ventana "Añadir Resultado" permitiendo la posibilidad de añadir un nuevo resultado a una prueba existente.
    - Nota 1: Ver requisitos correspondientes a la ventana "Añadir Resultado".
    - Nota 2: deberá seleccionarse una prueba sin resultado asociado. En caso contrario el mensaje de error "La prueba ya contiene un resultado asignado" será mostrado
    - Nota 3: En caso de que no se haya seleccionado ninguna Prueba aparecerá un mensaje de aviso mostrando: "Ninguna Prueba se ha seleccionado para la asociación".

- **Modificar Resultado:** Se abrirá la ventana "Modificar Resultado" permitiendo la posibilidad de modificar un resultado existente.
    - Nota 1: En caso de que la prueba seleccionada no contenga actualmente ningún resultado aparecerá un mensaje de aviso mostrando: "La prueba seleccionada no incluye un resultado".
    - Nota 2: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: "Ninguna prueba se ha seleccionado para asociar".
    - Nota 3: Ver requisitos correspondientes a la ventana "Modificar resultado".
- **Borrar Resultado:** permite la posibilidad de Borrar un resultado existente.
    - Nota 1: En caso de que no se haya seleccionado ningún requisito aparecerá un mensaje de aviso mostrando: "Ningún resultado se ha seleccionado para Borrar".
    - Nota 2: Ver requisito asociado -Requisito 5-.
- **Cerrar Ventana:** Se cerrará la ventana actual volviendo a la ventana "Menú principal".

Requisito 39: En el momento que se desee Borrar un resultado (pulsando el botón [Borrar resultado]), se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar el borrado del resultado seleccionado de la siguiente forma:

- **Sí:** Se borrará el resultado, eliminándolo de la base de datos.
- **No:** Se volverá a la ventana "Resultados" sin realizar ningún cambio.

Relativos a la ventana de Pruebas:

Requisito 40: Tras haber seleccionado la opción de Pruebas en el menú principal deberá mostrarse la ventana "Pruebas".

Requisito 41: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cerrar ventana].

Requisito 42: Al abrirse esta ventana mostrara una lista con todas las pruebas existentes en la base de datos así como dos cuadros de texto no editables donde se mostrara el input y el output de la prueba respectivamente, así como dos listas una donde se muestran los requisitos asociados y otra para las incidencias asociadas a cada prueba.

Requisito 43: La ventana "Pruebas" dispondrá de 7 botones disponibles: [Nueva Prueba], [Modificar Prueba], [Borrar Prueba], [Añadir Requisito], [Añadir incidencia], [Ver Prueba] y [Cerrar Ventana]. Dichos botones se comportaran de la siguiente manera:

- **Nueva Prueba:** Se abrirá la ventana “Añadir Prueba” permitiendo la posibilidad de añadir una nueva prueba a la base de datos.
    - Nota: Ver requisitos correspondientes a la ventana “Añadir Prueba”.
- **Modificar Prueba:** Se abrirá la ventana “Modificar Prueba” permitiendo la posibilidad de modificar una Prueba existente.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para modificar”.
    - Nota 2: Ver requisitos correspondientes a la ventana “Modificar Prueba”.
- **Borrar Prueba:** permite la posibilidad de Borrar una prueba existente.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para Borrar”.
    - Nota 2: Ver requisito asociado -Requisito 5-.
- **Añadir requisito:** Se abrirá la ventana “Añadir requisito” permitiendo la posibilidad de asociar un requisito a una prueba determinada.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para la asociación”.
    - Nota 2: Ver requisitos correspondientes a la ventana “Añadir requisito”.
- **Añadir incidencia:** Se abrirá la ventana “Añadir incidencia” permitiendo la posibilidad de asociar una incidencia a una prueba determinada.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para la asociación”.
    - Nota 2: Ver requisitos correspondientes a la ventana “Añadir incidencia”.
- **Borrar requisito:** Se borrara el requisito actual asociado a la prueba.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para la asociación”.
    - Nota 2: aparecerá un mensaje de confirmación del borrado.
- **Borrar Incidencia:** Se borrara La incidencia actual asociado a la prueba.
    - Nota 1: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para la asociación”.
    - Nota 2: aparecerá un mensaje de confirmación del borrado.
- **Ver Prueba:** Se mostrará el input de la prueba seleccionada en el correspondiente cuadro de texto y su output en el cuadro de texto colindante (en blanco en caso de no tener output).

  - Nota: En caso de que no se haya seleccionado ninguna prueba aparecerá un mensaje de aviso mostrando: “Ninguna prueba se ha seleccionado para mostrar”.
- **Cerrar Ventana:** Se cerrará la ventana actual volviendo a la ventana “Menú principal”.

Requisito 44: En el momento que se desee Borrar una prueba (pulsando el botón [Borrar prueba]), se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar el borrado de la prueba seleccionada de la siguiente forma:

- **Sí:** Se borrará la prueba, eliminándolo de la base de datos.
- **No:** Se volverá a la ventana “Pruebas” sin realizar ningún cambio.

Relativos a la ventana de Incidencias:

Requisito 45: Tras haber seleccionado la opción de Incidencias en el menú principal deberá mostrarse la ventana “Incidencias”.

Requisito 46: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cerrar ventana].

Requisito 47: Al abrirse esta ventana mostrara una lista de todas las incidencias existentes y un cuadro de texto disponible para mostrar la descripción de las incidencias.

Requisito 48: La ventana “Incidencias” dispondrá de 5 botones disponibles: [Ver Incidencia], [Nueva Incidencia], [Modificar Incidencia], [Borrar Incidencia] y [Cerrar Ventana]. Dichos botones se comportaran de la siguiente manera:

- **Ver Incidencia:** Se mostrará la descripción de la incidencia seleccionada en el correspondiente cuadro de texto.
  - Nota: En caso de que no se haya seleccionado ninguna incidencia aparecerá un mensaje de aviso mostrando: “Ninguna Incidencia se ha seleccionado para mostrar”.
- **Nuevo Incidencia:** Se abrirá la ventana “Nueva Incidencia” permitiendo la posibilidad de crear una nueva incidencia.
  - Nota: Ver requisitos correspondientes a la ventana “Nueva Incidencia”.
- **Modificar Incidencia:** Se abrirá la ventana “Modificar Incidencia” permitiendo la posibilidad de modificar una incidencia existente.
  - Nota 1: En caso de que no se haya seleccionado ninguna incidencia aparecerá un mensaje de aviso mostrando: “Ninguna incidencia se ha seleccionado para modificar”.
  - Nota 2: Ver requisitos correspondientes a la ventana “Modificar Incidencia”.
- **Borrar Incidencia:** permite la posibilidad de Borrar una incidencia existente.

  - Nota 1: En caso de que no se haya seleccionado ninguna incidencia aparecerá un mensaje de aviso mostrando: “Ninguna incidencia se ha seleccionado para Borrar”.
  - Nota 2: Ver requisito asociado -Requisito 5-.
- **Cerrar Ventana:** Se cerrará la ventana actual volviendo a la ventana “Menú principal”.

Requisito 49: En el momento que se desee Borrar una incidencia (pulsando el botón [Borrar incidencia]), se deberá mostrar un mensaje de advertencia dando la posibilidad de Aceptar o rechazar borrado de la incidencia seleccionada de la siguiente forma:

- **Sí:** Se borrará la incidencia, eliminándola de la base de datos.
- **No:** Se volverá a la ventana “Incidencia” sin realizar ningún cambio.

Requisito 50: Al pulsar doble click sobre la ID de una incidencia mostrada en la lista izquierda de la ventana, está función actuará como si pulsáramos el botón [Ver Incidencia]. De este modo su descripción se mostrará en el cuadro de texto correspondiente.

Relativos a las Ventanas derivadas de los Botones del Menú Principal:

Ventanas de creación:Añadir Requisito:

Requisito 51: Tras haber seleccionado la opción de nuevo requisito en la ventana “Requisitos” deberá mostrarse la ventana “Añadir requisitos”.

Requisito 52: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 53: Al abrirse esta ventana mostrara un cuadro de texto en blanco disponible para escribir la descripción del nuevo requisito.

Requisito 54: La ventana “Añadir requisitos” dispondrá de 2 botones disponibles: [Añadir Requisito] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Añadir Requisito:** guardará los cambios almacenados en el cuadro de texto como un nuevo requisito creado. Usará el último número disponible de la lista de requisitos ej.: REQ-010.
  - Nota 1: Si el cuadro de texto se deja vacío se mostrará un mensaje de error diciendo “El cuadro de texto está vacío”.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana “Requisitos” sin guardar los cambios.

Añadir Resultado:

Requisito 55: Tras haber seleccionado la opción de nuevo resultado en la ventana "Resultados" deberá mostrarse la ventana "Añadir Resultados".

Requisito 56: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisito 57: La ventana "Añadir Resultado" dispondrá de una lista desplegable mostrando las opciones pasado y fallado donde se seleccionará el tipo de resultado que se quiere añadir. Y 2 botones disponibles: [Añadir Resultado] y [Cancelar] y Dichos botones se comportaran de la siguiente manera:

- **Añadir Resultado:** guardará los cambios seleccionados de la lista desplegable en la prueba previamente seleccionada.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Resultados" sin guardar los cambios.

Añadir Prueba:

Requisito 58: Tras haber seleccionado la opción de nueva prueba en la ventana "Pruebas" deberá mostrarse la ventana "Añadir Prueba".

Requisito 59: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 60: Al abrirse esta ventana mostrara dos cuadros de texto en blanco disponible para escribir el input y el output de la nueva prueba.

Requisito 61: La ventana "Añadir Prueba" dispondrá de 2 botones disponibles: [Añadir Prueba] y [Cancelar] y Dichos botones se comportaran de la siguiente manera:

- **Añadir Prueba:** guardará los cambios almacenados en los cuadros de texto como una nueva prueba creada. Usará el último número disponible de la lista de pruebas ej.: TEST-010.
- Nota 1: El campo output podrá dejarse en blanco. No obstante, el campo input se deberá rellenar obligatoriamente. En caso contrario el mensaje de error "El cuadro de texto está vacío" será mostrado.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Pruebas" sin guardar los cambios.

Añadir Incidencia:

Requisito 62: Tras haber seleccionado la opción de Nueva incidencia en la ventana "Incidencias" deberá mostrarse la ventana "Añadir incidencia".

Requisito 63: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 64: Al abrirse esta ventana mostrara un cuadro de texto en blanco disponible para escribir la descripción del nuevo requisito.

Requisito 65: La ventana "Añadir incidencia" dispondrá de 2 botones disponibles: [Añadir incidencia] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Añadir Incidencia:** guardará los cambios almacenados en el cuadro de texto como una nueva incidencia creada. Usará el último número disponible de la lista de incidencias ej.: INC-010.
  - Nota 1: Si el cuadro de texto se deja vacío se mostrará un mensaje de error diciendo "El cuadro de texto está vacío".
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Incidencias" sin guardar los cambios.

Ventanas de modificación:

Modificar Requisito:

Requisito 66: Tras haber seleccionado la opción de modificar requisito en la ventana "Requisitos" deberá mostrarse la ventana "Modificar requisitos".

Requisito 67: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 68: Al abrirse esta ventana mostrara dos cuadros de texto, uno mostrando la ID de la prueba seleccionada y el otro con la anterior descripción del requisito disponible para efectuar modificaciones.

Requisito 69: La ventana "Modificar requisitos" dispondrá de 2 botones disponibles: [Modificar Requisito] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Modificar Requisito:** guardará los cambios almacenados en el cuadro de texto como la nueva descripción del requisito seleccionado.
  - *Nota 1:* Si el cuadro de texto se deja vacío se mostrará un mensaje de error diciendo "El cuadro de texto está vacío".
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Requisitos" sin guardar los cambios.

Modificar Resultado:

Requisito 70: Tras haber seleccionado la opción de modificar resultado en la ventana "Resultados" deberá mostrarse la ventana "Modificar Resultados".

Requisito 71: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisito 72: La ventana "Modificar Resultado" dispondrá de una lista desplegable mostrando las opciones pasado y fallado donde se seleccionará el tipo de resultado que se quiere seleccionar. Y 2 botones disponibles: [Modificar Resultado] y [Cancelar] y Dichos botones se comportaran de la siguiente manera:

- **Modificar Resultado:** guardará los cambios seleccionados de la lista desplegable en la prueba previamente seleccionada.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Resultados" sin guardar los cambios.

Modificar Prueba:

Requisito 73: Tras haber seleccionado la opción de modificar prueba en la ventana "Pruebas" deberá mostrarse la ventana "Modificar Prueba".

Requisito 74: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 75: Al abrirse esta ventana mostrara tres cuadros de texto, uno con la ID de la prueba y los otros dos con los valores de la prueba seleccionada (input y output) accesibles para escribir el nuevo input y el output de la misma.

Requisito 76: La ventana "Modificar Prueba" dispondrá de 2 botones disponibles: [Modificar Prueba] y [Cancelar] y Dichos botones se comportaran de la siguiente manera:

- **Modificar Prueba:** guardará los cambios almacenados en los cuadros de texto como modificación de la prueba seleccionada.
- **Nota 1**: El campo output podrá dejarse en blanco. No obstante, el campo input se deberá rellenar obligatoriamente. En caso contrario el mensaje de error "El cuadro de texto está vacío" será mostrado.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Pruebas" sin guardar los cambios.

Modificar Incidencia:

Requisito 77: Tras haber seleccionado la opción de modificar incidencia en la ventana "Incidencias" deberá mostrarse la ventana "Modificar Incidencias".

Requisito 78: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 79: Al abrirse esta ventana mostrara dos cuadros de texto, uno mostrando la ID de la prueba seleccionada y el otro con la anterior descripción de la incidencia disponible para efectuar modificaciones.

Requisito 80: La ventana "Modificar Incidencias" dispondrá de 2 botones disponibles: [Modificar Incidencia] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Modificar Incidencia:** guardará los cambios almacenados en el cuadro de texto como la nueva descripción de la incidencia seleccionada.
  - Nota 1: Si el cuadro de texto se deja vacío se mostrará un mensaje de error diciendo "El cuadro de texto está vacío".
- **Cancelar:** Cancela la acción actual. Volverá a la ventana "Requisitos" sin guardar los cambios.

Ventanas de Asociación:

Asociar Requisito:

Requisito 81: Tras haber seleccionado la opción de Añadir requisito en la ventana "Pruebas" deberá mostrarse la ventana "Asociar Requisito".

Requisito 82: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 83: Al abrirse esta ventana mostrara una lista con todos los requisitos posibles para ser asociados a la prueba.

Requisito 84: La ventana "Asociar requisito" dispondrá de 2 botones disponibles: [Añadir Requisito] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Añadir Incidencia:** guardará el requisito seleccionado de la lista en el campo correspondiente de la prueba seleccionada.

  - Nota 1: Si no se selecciona ningún requisito el mensaje de error “Ningún requisito se ha seleccionado para la asociación” será mostrado y se cerrará la ventana actual, volviendo a la ventana “Pruebas” sin guardar los cambios.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana “Pruebas” sin guardar los cambios.

Asociar Incidencia:

Requisito 85: Tras haber seleccionado la opción de Añadir incidencia en la ventana “Pruebas” deberá mostrarse la ventana “Asociar Incidencia”.

Requisito 86: Dicha ventana podrá cerrarse en cualquier momento mediante el botón CERRAR propio de la ventana o mediante el botón [Cancelar].

Requisitos 87: Al abrirse esta ventana mostrara una lista con todas las incidencias posibles para ser asociadas a la prueba.

Requisito 88: La ventana “Asociar incidencia” dispondrá de 2 botones disponibles: [Añadir Requisito] y [Cancelar]. Dichos botones se comportaran de la siguiente manera:

- **Asociar Incidencia:** guardará la incidencia seleccionada de la lista en el campo correspondiente de la prueba seleccionada.
  - Nota 1: Si no se selecciona ninguna incidencia el mensaje de error “Ninguna incidencia se ha seleccionado para la asociación” será mostrado y se cerrará la ventana actual, volviendo a la ventana “Pruebas” sin guardar los cambios.
- **Cancelar:** Cancela la acción actual. Volverá a la ventana “Pruebas” sin guardar los cambios.

5.3 Estudio de lenguajes de programación y/o entornos para el desarrollo de esta herramienta. Justificación técnica.

5.3.1 Respecto al lenguaje de programación.

Respecto al lenguaje de programación usado, existía la posibilidad de usar C++ o Java debido a que estos lenguajes son más conocidos y han sido aprendidos durante el estudio del Grado de Ingeniería de Computadores. No obstante, al final se decidió usar Java por

comodidad, debido a que parece un lenguaje más intuitivo, ofrece más librerías disponibles y documentación a mano. Además, es el lenguaje que más se ha usado en los últimos años en aplicativos software de desarrollo particular. Además el entorno que se usará para Java contiene integrados el driver y las librerías para la base de datos utilizada lo cual hace más fácil el acceso a un entorno de desarrollo integrado de forma accesible.

Para la Base de Datos existían las posibilidades de usar alguna gestor orientado a SQL o de la sugerencia aportada por la tutora de este proyecto de usar MongoDB, un tipo de NoSQL que actualmente está cogiendo mucho impulso, y que se puede adaptar fácilmente al desarrollo de este proyecto dado que a simple vista, el diseño de la base de datos que requiere la herramienta NovaTest no es complicado en términos de las relaciones existentes entre entidades. Debido a que esta podía ser una buena oportunidad para aumentar los conocimientos y aprender una nueva herramienta actual se decidió usar MongoDB. Estas bases de datos son muy fáciles de desarrollarse en Java, ya que como se ha mencionado anteriormente los entornos de desarrollo de Java permiten añadir el driver de MongoDB y usar las librerías creadas.

5.3.2 Respecto al entorno de desarrollo.

Para este apartado ha de mencionarse que la posibilidad de entornos de desarrollo con los que se pueden trabajar para Java son muchos, pero por simplicidad y por resumir el apartado hablaremos directamente del entorno de desarrollo utilizado.

Netbeans 8.0 ha sido el entorno de desarrollo utilizado para la creación de la herramienta software debido principalmente a que a la hora de crear la GUI (interfaz gráfica) con Netbeans es mucho más simple. Dado que no se tenía mucho conocimiento y fluidez a la hora de crear interfaces gráficas, y menos en Java, Netbeans facilitaba este "punto débil" a la hora del desarrollo. Esto permitiría avanzar rápidamente con la programación a la vez que la interfaz creada no sería compleja y tediosa de utilizar, ya que a la vez que sencillo de crear, la interfaz creada también sería simple visualmente y en su uso debido a las opciones "limitadas" que da Neatbeans para incluir en el software sin tener que tocar directamente el código de creación de ventanas, botones, listas, etc.

Como conclusión, debido a que el software tenía destinado un uso a nivel educativo universitario y por tanto debía ser fácil de usar, comprender y no debía tener una apariencia abrumadora visualmente, está en concordancia con las opciones que Neatbeans nos ofrecía. Además, la estructuración del proyecto en Netbeans y la forma de incluir el driver de MongoDB es mucho más intuitiva que en otros entornos, como eclipse por ejemplo. Otro añadido es que el ejecutable creado puede ser ejecutado en cualquier ordenador y además es fácil de mantener y reutilizar, ya que Netbeans tiene una interfaz muy intuitiva.

5.4 Diseño del plan de pruebas.

5.4.1 Resumen Ejecutivo

El siguiente plan describirá en qué consistirá exactamente el plan de pruebas de la herramienta software. En este se definirá el alcance de las pruebas, las herramientas necesarias para su correcto funcionamiento y los criterios para decidir su aprobación o rechazo.

Este plan de pruebas deberá ejecutarse durante y después de haber desarrollado el software. Su alcance será probar cada módulo individualmente así como la herramienta en conjunto y las pruebas unitarias que sean necesarias. A continuación se explicará cada uno de estos aspectos más en detalle.

5.4.2 Alcance de las pruebas

En este apartado se describirá a que elementos, funciones deben ser probados así como otros requisitos no deberán ser probados. Además también se indicará la estrategia que se seguirá para la ejecución de este plan de pruebas. Para ello se ha dividido en cuatro apartados distintos y en cada uno se describirá uno de estos aspectos.

5.4.2.1 Elementos de Pruebas

Los módulos o componentes que se deberán probar de la herramienta software son los siguientes:

- Ventana Inicial
- Ventana Menú principal
- Ventana Requisitos
- Ventana Pruebas
- Ventana Resultados
- Ventana Incidencias
- Ventanas derivadas de las principales

De esta forma la herramienta quedará probada de forma modular, cada módulo independiente uno de otro. Esto se llevará a cabo con pruebas unitarias para cada módulo. Para probar la funcionalidad de la herramienta en conjunto también deberán desarrollarse otras pruebas de sistema para comprobar que todos ellos son capaces de funcionar simultáneamente sin colisiones ni errores.

5.4.2.2 Funcionalidades a probar

Para comprobar correctamente que la herramienta funcione las siguientes funcionalidades deberán ser validadas satisfactoriamente:

- Continuar con una Base de datos existente.
- Crear una nueva base de datos
- Funcionalidades de visualización (Pruebas, requisitos e incidencias).
- Funcionalidades de creación (Pruebas, requisitos, incidencias y resultados).
- Funcionalidades de modificación (Pruebas, requisitos, incidencias y resultados).
- Funcionalidades de borrado (Pruebas, requisitos, incidencias y resultados).
- Funcionalidades de Asociación de Requisitos e Incidencias (Pruebas)
- Funcionalidades de cierre de ventana (Pruebas, requisitos, incidencias y resultados)
- Funcionalidades de salvado de datos de ventanas derivadas de las principales (Añadir y modificar Pruebas, requisitos, incidencias y resultados).
- Funcionalidades de cancelación de salvado de datos de ventanas derivadas de las principales (Cancelar para Añadir y modificar Pruebas, requisitos, incidencias y resultados)

Para ello se llevarán a cabo pruebas unitarias de forma análoga al alcance de pruebas modulares, siendo en este caso, más específicos con lo que se requiere probar: la funcionalidad.

5.4.2.3 Elementos a NO probar

En este apartado se decidirá aquello que no será necesario probar para que entre dentro del alcance de la aceptación. En este caso en concreto, aquello que no será necesario probar son los requisitos NO FUNCIONALES, serán dados por hecho que funcionan y son validados de una forma correcta. Esto es debido a que probar estos requisitos quitaría tiempo para el resto de pruebas y no son requisitos dependientes del software, es decir, son genéricos y se consideran inmersos en el mismo, van implícitos. En caso de que alguno fallara el riesgo asumido es leve, ya que las modificaciones pertinentes no afectarían al funcionamiento del software en sí mismo.

5.4.2.4 Enfoque de Pruebas

La mayoría de las pruebas serán del tipo unitarias, ya que con ellas se probarán las funcionalidades y módulos de forma independiente. No obstante, también deberá haber pruebas funcionales de sistema para comprobar el conjunto de módulos en sí mismo, es decir, que todos los módulos juntos funcionan como se espera.

Además se harán pruebas de rendimiento y/o estabilidad para comprobar que el sistema no se cierra de forma inesperada así como otras pruebas como pueden ser el correcto funcionamiento de la base de datos de MongoDB.

5.4.2.5 Criterios de aceptación o rechazo

A continuación se expondrán aquellas pautas que deberán seguirse para considerar el software maduro y estable y por tanto apto para su uso.

5.4.3 Criterios de aceptación

Los siguientes criterios deberán ser conseguidos para considerar los requisitos de la herramienta software por satisfechos:

- 100% de pruebas unitarias ejecutadas.
- 90% de pruebas unitarias pasadas.
- 100% de pruebas funcionales ejecutadas
- 85% de pruebas funcionales pasadas
- La herramienta software está disponible y estable el 90% o más del tiempo de uso.
- No habrá más de 20 incidencias abiertas para el software.
- La base de datos se mantendrá estable en todo momento en el que la herramienta esté usándose.

Una vez superados todos estos aspectos se podrá considerar el proyecto como satisfactorio y por tanto podrá ser puesto en producción bajo mantenimiento por la aparición de posibles errores no descubiertos durante la ejecución de este plan de pruebas.

5.4.3.1 Criterios de rechazo

En el momento en que uno o más de los aspectos mencionados en el apartado anterior no sean satisfechos, el software no podrá seguir adelante

en su fase de producción y por tanto deberá volver a desarrollarse para paliar el defecto que se hubiera encontrado. En caso de que 3 o más de estos aspectos fallaran, el software y los requisitos deberían volver a ser llevados a la fase de especificación para comprobar que los fallos no han sido a la hora de diseñarlo o de mal entendimiento de los requisitos.

Una vez estos aspectos hayan sido corregidos el software deberá volver a pasar otro plan de pruebas para poderse dar por validado.

5.4.4 Requerimientos de Entornos y Herramientas requeridas

Las herramientas necesarias para la correcta ejecución del plan de pruebas serán las listadas a continuación, **todas ellas** deberán estar presentes a la hora de ejecutar el plan de pruebas:

- Un ordenador con sistema operativo Windows.
- Versión de java correctamente actualizada en la máquina de pruebas.
- Carpeta para arrancar la base de datos de MongoDB (incluida en la misma carpeta que el ejecutable de la herramienta).
- Ejecutable de la herramienta software NOVATests (no pudiendo usarse el proyecto del entorno Netbeans)
- Driver de MongoDB (incluido en la carpeta de la herramienta).

En caso de que alguna de ellas faltara, el plan de pruebas no podría dar comienzo.

6. Diseño

6.1 Diseño de la herramienta software

En los siguientes apartados se describirá brevemente como está diseñada la herramienta software en tres niveles o capas diferentes: lógico, de acceso de datos y de interfaz (GUI). A continuación se adjuntarán algunos diagramas UML (incluyendo el diagrama de clases) para un mejor entendimiento de la herramienta. El nivel lógico representa las clases y elementos propios del programa que interactúan para unir interfaz y DB. La interfaz es con la que opera el usuario y consta de ventanas, botones y cuadros de texto en su mayoría. Por último, el nivel de acceso de datos estará compuesto por la base de datos y todos sus elementos.

6.1.1 Nivel Lógico

El nivel lógico se fundamenta principalmente en el uso del driver de MongoDB para el uso de las funciones pertinentes para realizar las operaciones necesarias para la herramienta. Entre algunas de estas funciones pueden destacarse: Creación de la base de datos, Creación de las colecciones, Creación de documentos, Modificación de documentos, etc.

En este nivel también podemos incluir las funciones que relacionan cada elemento y los métodos que llaman a la clase principal en la que se fundamenta el programa (clase NOVATests). Desde esta clase, se añade, modifica o borra cualquier documento de la base de datos. Esta clase constituye el punto de inflexión para todos los demás elementos estáticos del sistema. Aunque cada elemento tenga sus propios métodos, en ninguna de ellas se interacciona con la base de datos, simplemente hacen comprobación de errores.

Este nivel nos permite independencia entre la interfaz y el nivel de Acceso de datos, de forma que se pueda modificar el uno sin que el otro se vea afectado. Simplemente haciendo cambios a nivel lógico se podría adaptar los otros dos niveles.

6.1.2 Nivel de Acceso de Datos

Este nivel va a utilizar una base de datos de MongoDB, es decir, noSQL. Ésta será la que contenga toda la información de NOVATests. Para ello sus colecciones y documentos estarán estructurados de una manera ordenada, eficiente y coherente.

La Base de Datos inicial contendrá cuatro colecciones vacías en un principio (sin documentos creados): Requisitos, Incidencias, Resultados y Pruebas. Para esta herramienta no existe restricción en cuanto a autorización se refiere, no será

necesario un log in. Todos los usuarios serán iguales desde el momento en que se abra la herramienta. Todos los usuarios tendrán acceso a la misma funcionalidad.

Cada una de las colecciones tendrá sus atributos propios. Para el modelo que se ha seguido (mostrado a continuación) se ha hecho una relación N:1 es decir, varios a uno. Esto significa que la colección Pruebas estará compuesta y se relacionará con el resto de tablas de forma que Pruebas tendrá Requisitos, Resultados y puede contener Incidencias.

6.1.3 Diseño GUI

En la siguiente imagen se puede encontrar un ejemplo de la interfaz gráfica de la herramienta software NOVATests. Todas las ventanas siguen un patrón común aunque cada una de ellas puede incluir distinta información en sus elementos.

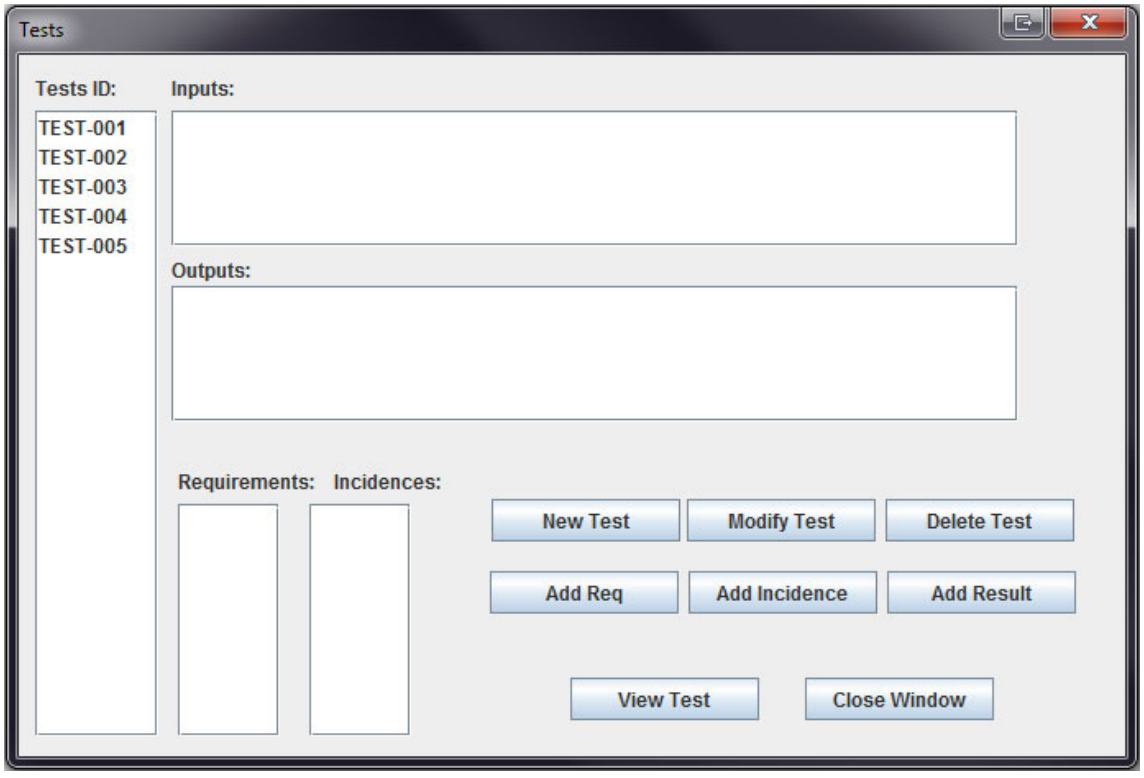

Figura 11: Ventana Pruebas

Para el diseño de la interfaz gráfica y por el uso didáctico que va a tener la herramienta, siempre se ha buscado la simplicidad de uso, el fácil entendimiento y que sea intuitivita. Por ello no consta de muchos elementos y mucha diversidad de colores y ventanas, todas son parecidas para dar una sensación de familiaridad y siguen el mismo patrón a la hora de colocación de elementos para que no sea confuso.

Dicha colocación puede resumirse de la siguiente manera:

- Una cabecera con el título de la ventana en la que nos encontramos actualmente. De este modo nunca estaremos perdido y siempre sabremos en que parte de la herramienta estamos trabajando.
- Listas y cuadros de texto que muestran diversa información como IDs, descripciones, inputs, resultados, etc. Dichos cuadros siempre estarán ubicados de la parte izquierda y empezando por la parte superior de la ventana, de modo que será lo primero que encontremos una vez abramos la ventana, así podremos visualizar la información con un simple golpe de vista.
- Botones que realizan las funciones necesarias para cada ventana de la herramienta. Dichos botones estarán situados en la parte inferior derecha de las ventanas para cada una. Esta colocación permite un fácil uso de los botones sin necesidad de alejar mucho la vista ni el ratón unos de otros, están todos al alcance y ubicados siempre en las mismas zonas.
- Etiquetas describiendo la información contenida en los distintos cuadros de texto. Cada etiqueta estará colocada encima de cada cuadro de texto.

1 Como podemos observar no hay muchos elementos incluidos dentro de la herramienta para no saturarla y que no sea compleja. La información referente a cada ventana, a la interfaz y a las funciones de cada botón e información mostrada en los cuadros de texto será ampliada en el manual de usuario adjunto en este documento (punto 12. Anexo: Manual de Usuario).

6.2 Diagramas UML

Este diagrama será mostrado en las siguientes dos hojas debido a su tamaño.

Figura 12: Diagrama UML completo

7. Construcción e Implementación del Entorno Software

7.1 Tecnologías y Herramientas

Java y Netbeans 8.0:

*"**Java** es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible." [4].* Al ser Java un lenguaje WORA (write once, run anywhere) se adapta perfectamente a las necesidades de este proyecto. Lo que WORA quiere decir es que una vez lo compiles podrás ejecutarlo en cualquier otra máquina sin necesidad de recompilarlo. Además es un lenguaje que minimiza las dependencias de implementación tanto como sea posible, factor que ayuda a que sea simple y fácil de usar como se persigue en los objetivos principales de este proyecto.

En cuanto a Netbeans 8.0, se ha elegido como entorno de desarrollo para la implementación de la herramienta de software propuesta en este Proyecto de Fin de Grado. Netbeans es un entorno de desarrollo de dominio público y libre de uso. Permite que las aplicaciones sean desarrolladas en base de módulos. Dichos módulos hacen parte de un archivo Java que contiene otras clases java que se comunican con las APIs de Netbeans. Estas aplicaciones pueden ser extendidas con nuevos módulos. Netbeans IDE soporta todos los tipos de programación en Java (p.e. concurrente, orientado a la web, orientado a tecnologías móviles). La modularidad y simplicidad a la hora de codificar en Java, y la facilidad de Netbeans como entorno de desarrollo es lo que ha hecho que elijamos este producto para nuestro proyecto.

MongoDB y MongoDB Driver:

"Un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto." [5]. Lo que significa NoSQL es, que en lugar de guardar los datos en tablas, se guarda en estructuras de datos tipo JSON (BSON en el caso de Mongo). Es gratis y de libre distribución, lo cual lo hace muy accesible. Además con un sistema NoSQL se simplifica mucho la herramienta debido a que ya no se tiene que estar con relaciones de tablas tediosas, si no que ahora el acceso a datos es mucho más intuitivo y rápido. No obstante, una de los inconvenientes que se le atribuyen a NoSQL es que solo es viable para volúmenes de datos poco relacionados. En el caso de este proyecto en particular, viene bien utilizarlo debido que no tenemos un diseño de base de datos complejo. El driver de Mongo DB es una librería que contiene todas las funciones necesarias para conseguir ese acceso y modificación de los datos almacenados en las

estructuras BSON así como la creación de la Base de Datos y de las colecciones y documentos necesarios.

7.2 Diagrama de Navegación

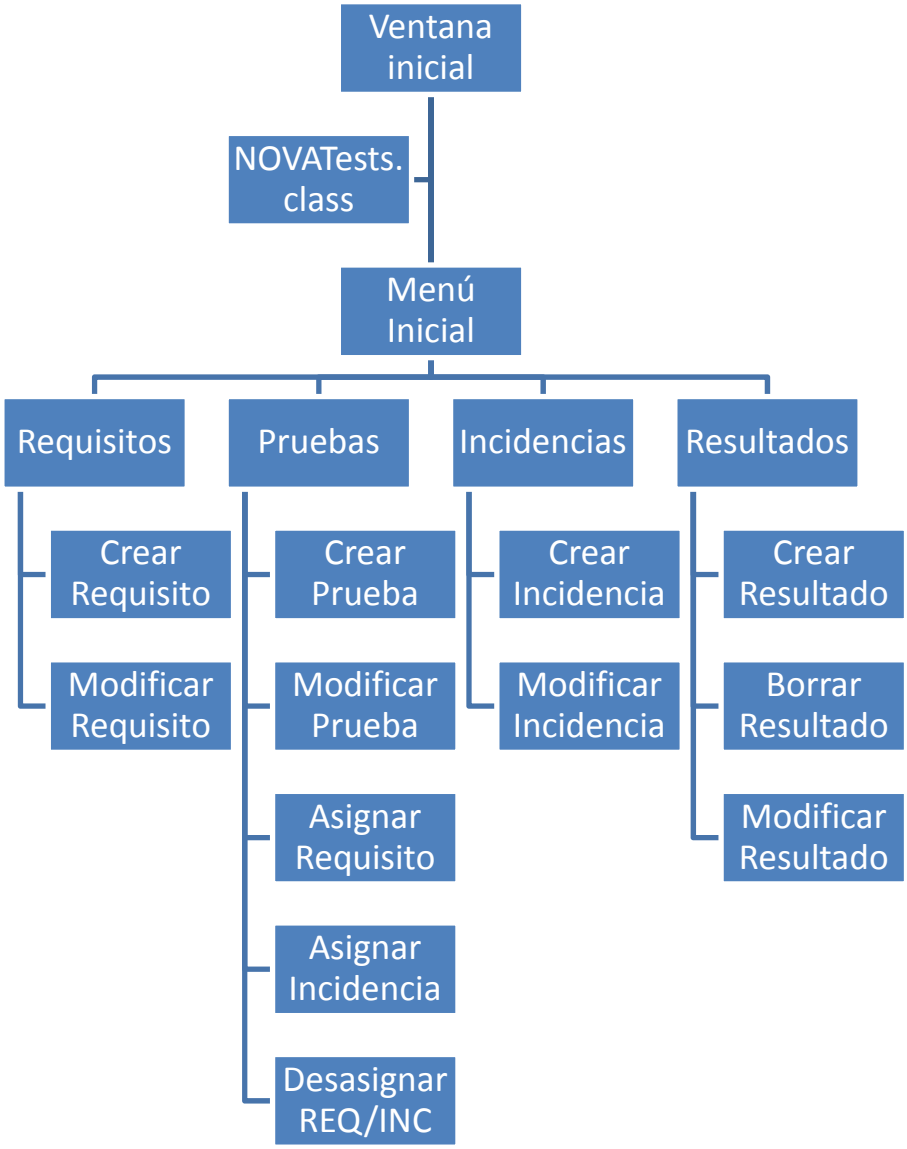

Figura 13: Diagrama de Navegación.

7.3 Interfaz de Usuario y Uso de la Herramienta

En este apartado se explicará el funcionamiento y modo de uso de la interfaz y la herramienta. Para ello se usarán una serie de imágenes y fotografías que se detallarán en cada caso como convengan.

Inicializaciones previas:

Previo uso de la herramienta debemos configurar y abrir la base de datos de MongoDB. Para ello usaremos el archivo [Startmongo.bat] haciendo doble click, con esto la base de datos ya estará lista para introducir y buscar datos.

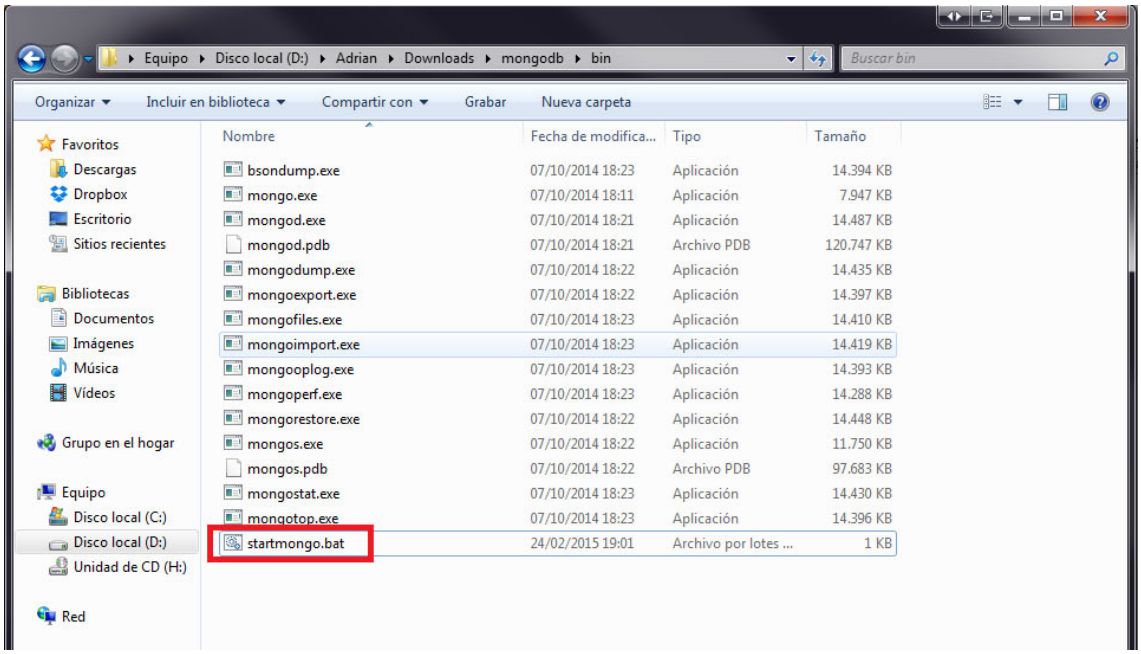

Figura 14: Archivo de inicialización de MongoDB

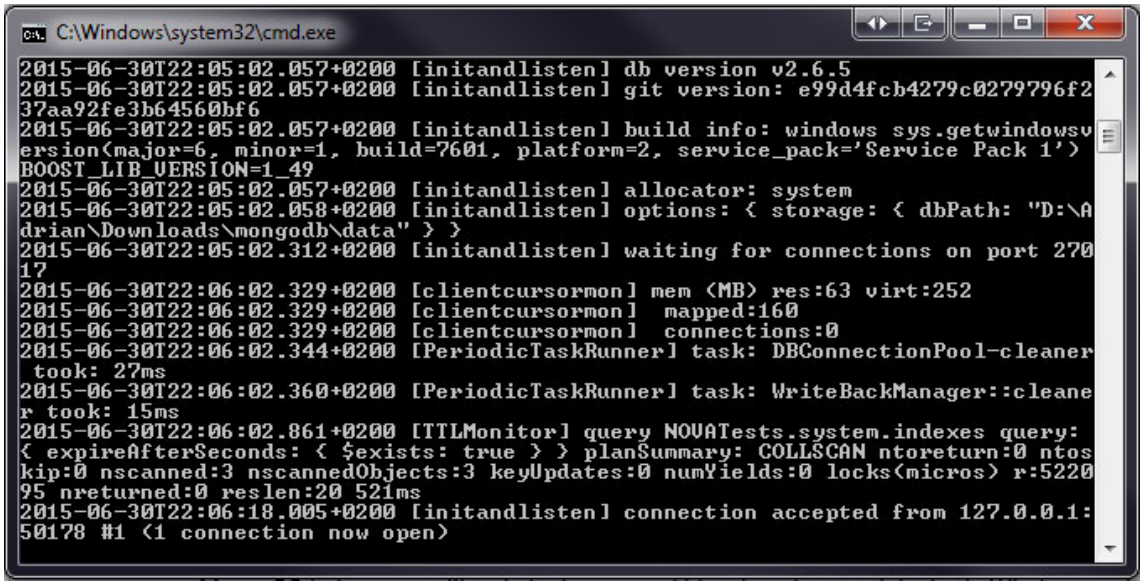

Figura 15: Correcto funcionamiento de MongoDB

En caso de que queramos cambiar la ruta de la base de datos (por defecto está guardado D:\Adrian\Downloads\mongodb\data) debemos abrir el MongoDB.bat con un editor de texto, como el bloc de notas por defecto de Windows o el notepad. Una vez abierto, en donde está especificada la ruta la cambiaremos a la que deseemos. Esto también nos permite almacenar varias bases de datos en distintos directorios sin borrarlas, pero deberemos ejecutar estos pasos cada vez que queramos iniciar la herramienta.

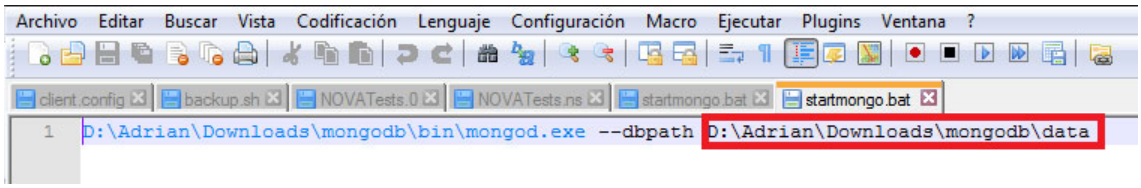

Figura 16: ruta del archivo Startmongo.bat

Ventana Inicial del programa:

Esta ventana será la que nos encontraremos en cada apertura del programa por medio del ejecutable. Cualquiera de las opciones que seleccionemos nos llevara siempre a la ventana del Menú Principal, no obstante cada botón realiza una función distinta.

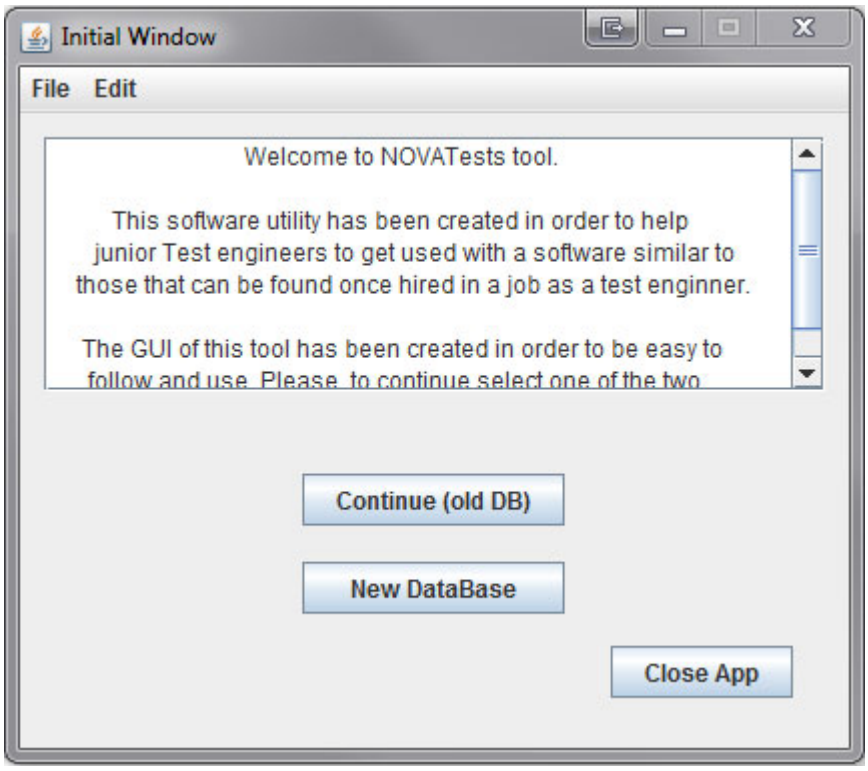

Figura 17: Ventana inicial.

Pulsando el botón de Continuar con la Base de datos, usará la base de datos tal como está actualmente, sin ningún cambio respecto a la última vez que abrimos el programa. Esta es la opción que se usará normalmente para continuar con el trabajo que estábamos haciendo. Asegurarse de que la base de datos de MongoDB está ejecutándose.

Mediante el botón de Nueva Base de Datos la actual base de datos se borrará y nos dejará una totalmente limpia, para iniciar un nuevo proyecto por ejemplo. En cualquier caso y como ya hemos mencionado, cualquier opción nos llevará a la ventana del Menú Principal.

El botón de salir de la aplicación cerrará la herramienta manteniendo los cambios almacenados en la base de datos.

Ventana Menú Principal:

Esta ventana será la más usada del programa, es la que nos da acceso a toda la funcionalidad de la herramienta. En ella podemos encontrar 4 opciones:

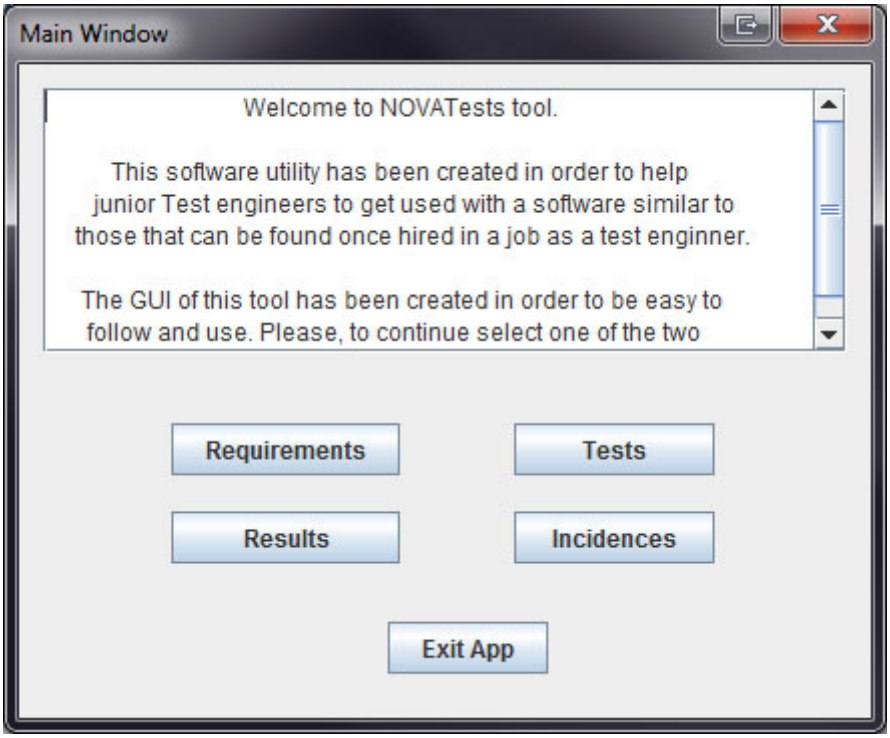

Figura 18: Ventana Principal.

A continuación explicaremos cada una de ellas:

- *Requisitos*
- *Incidencias*
- *Pruebas*
- *Resultados*

Ventana Requisitos:

Desde esta ventana podremos administrar la creación, modificación y borrado de requisitos. Para ello tenemos cada uno de los botones específicos para realizar cada función.

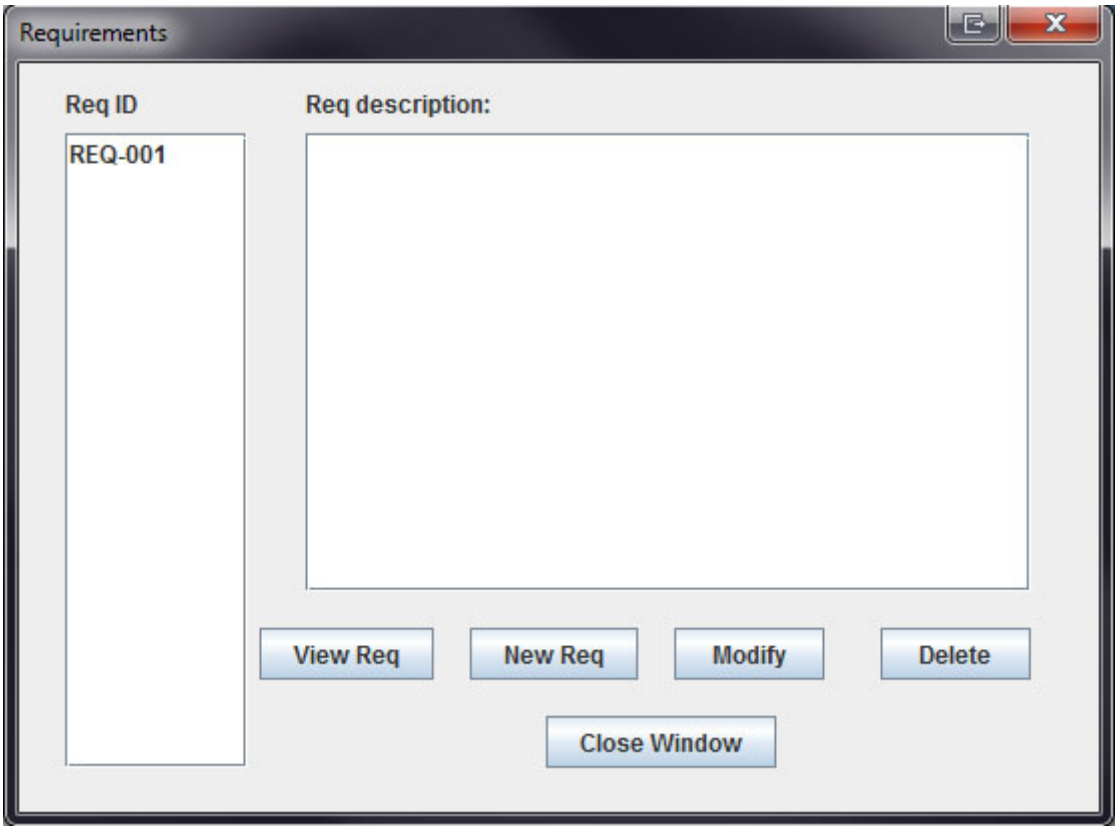

Figura 19: Ventana requisitos.

En la lista de la izquierda se mostrarán todos los requisitos de la base de datos, ordenados por ID. En el cuadro que hay en la parte derecha-superior se mostrará la descripción del requisito seleccionado. Los botones realizarán cada uno su función.

Pulsando el botón de “Ver requisito” y habiendo seleccionado previamente un requisito (de otra forma nos mostrará un mensaje de error diciendo que no hemos seleccionado ninguno) se mostrará en la caja de descripción, el contenido del requisito seleccionado. Esta función también puede ser invocada haciendo doble click sobre un requisito de la lista de la izquierda.

Mediante el botón “Crear Requisito” podremos añadir un nuevo requisito a la base de datos. Para ello al pulsarlo se nos mostrará la siguiente ventana:

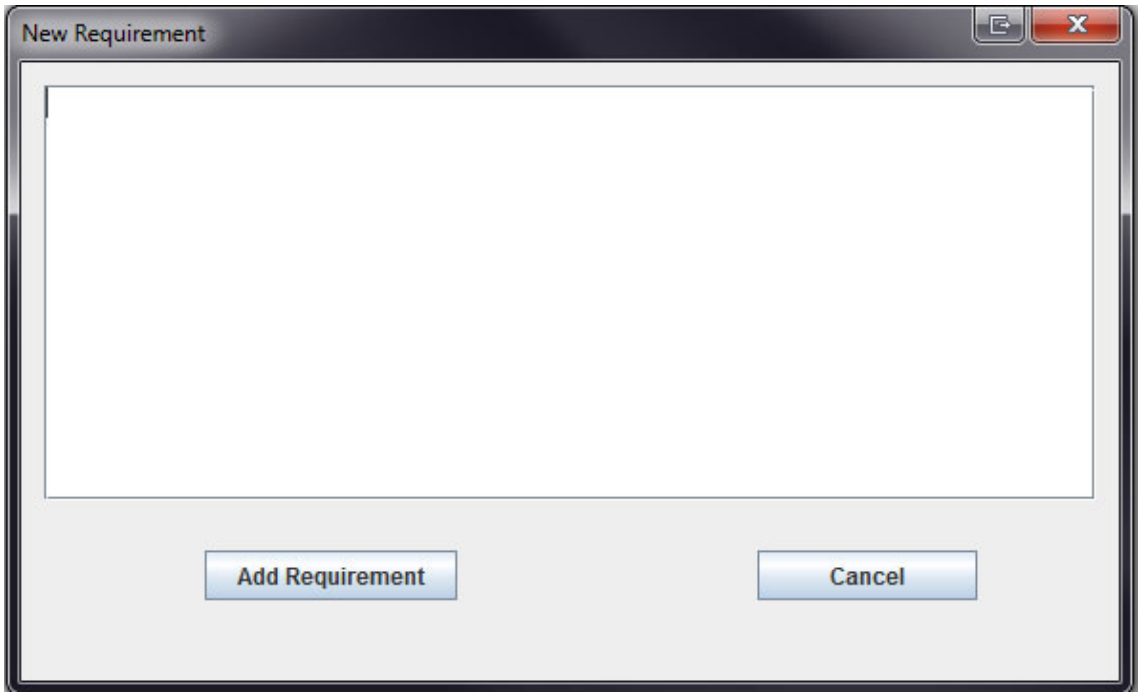

Figura 20: Ventana de nuevo requisito.

Desde aquí, una vez rellenemos el cuadro de texto (sino mostrará un error de que está vacía al aceptar) podremos guardar los cambios y crear el requisito con el contenido del cuadro de texto pulsando el botón añadir requisito. En cualquier otro caso se cancelará la creación del requisito si pulsamos cancelar o cerramos la ventana.

Si pulsamos el botón "Modificar requisito" y habiendo seleccionado previamente un requisito de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguno) se mostrará la siguiente ventana:

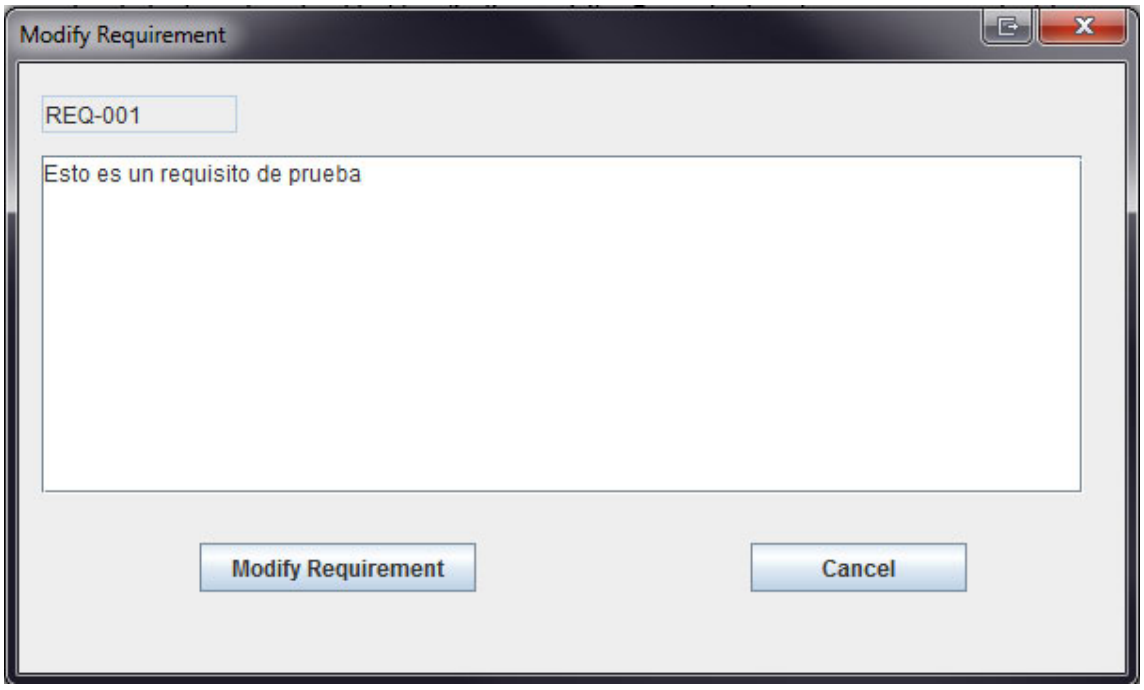

Figura 21: Ventana de modificar requisito.

En ella podemos observar la ID del requisito seleccionado y un cuadro de texto con la descripción actual con posibilidad de ser modificada. Una vez hayamos realizado los cambios pertinentes podremos guardarlos pulsando el botón Aceptar. En cualquier otro caso se cancelará la modificación del requisito si pulsamos cancelar o cerramos la ventana.

Por último si pulsamos el botón "Borrar requisito" y habiendo seleccionado previamente un requisito de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguno) se nos mostrará el siguiente cuadro de confirmación:

Figura 22: Mensaje de borrado de requisitos.

Desde aquí se podrá borrar el requisito pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando la ventana de confirmación.

Ventana Incidencias:

Desde esta ventana podremos administrar la creación, modificación y borrado de Incidencias. Para ello tenemos cada uno de los botones específicos para realizar cada función.

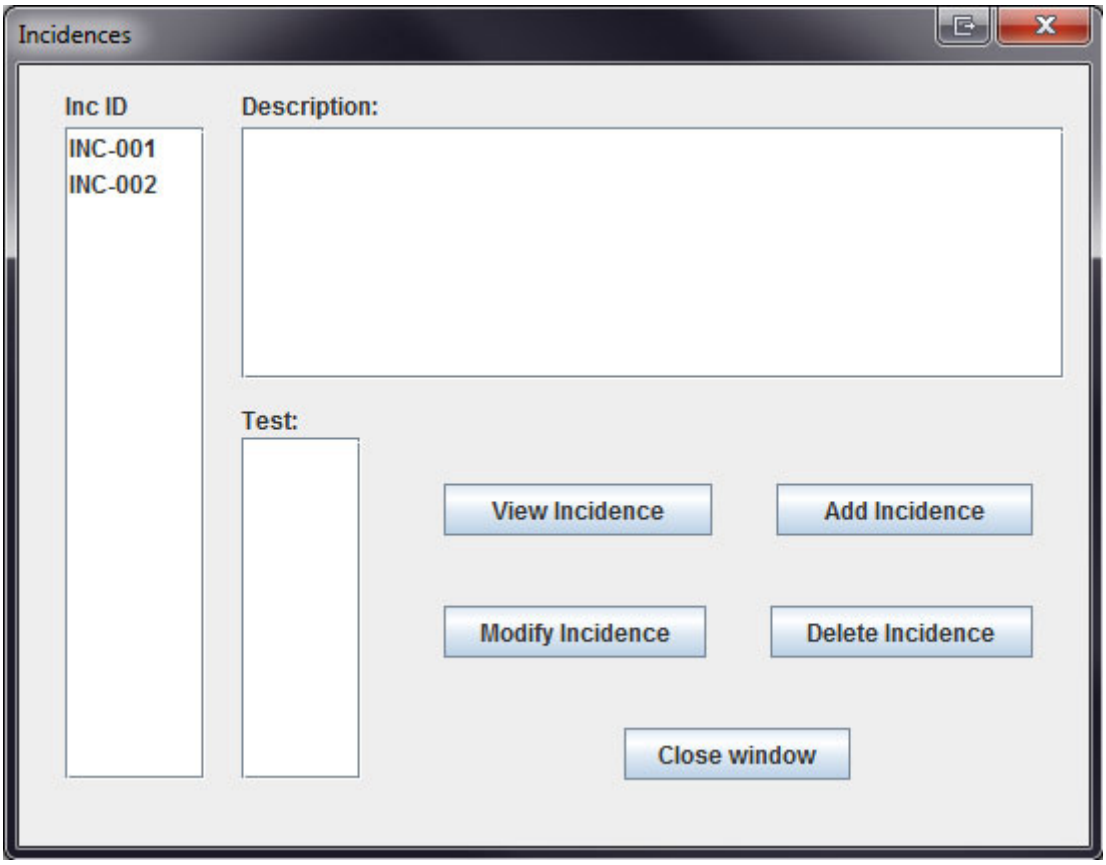

Figura 23: Ventana Incidencias.

En la lista de la izquierda se mostrarán todas las incidencias de la base de datos, ordenados por ID. En el cuadro que hay en la parte derecha-superior se mostrará la descripción de la Incidencia seleccionada. Los botones realizarán cada uno su función.

Pulsando el botón de "Ver Incidencia" y habiendo seleccionado previamente una incidencia (de otra forma nos mostrará un mensaje de error diciendo que no hemos seleccionado ninguna) se mostrará en la caja de descripción el contenido de la incidencia seleccionada. Esta función también puede ser invocada haciendo doble click sobre una incidencia de la lista de la izquierda.

Mediante el botón "Crear Incidencia" podremos añadir una nueva incidencia a la base de datos. Para ello al pulsarlo se nos mostrará la siguiente ventana:

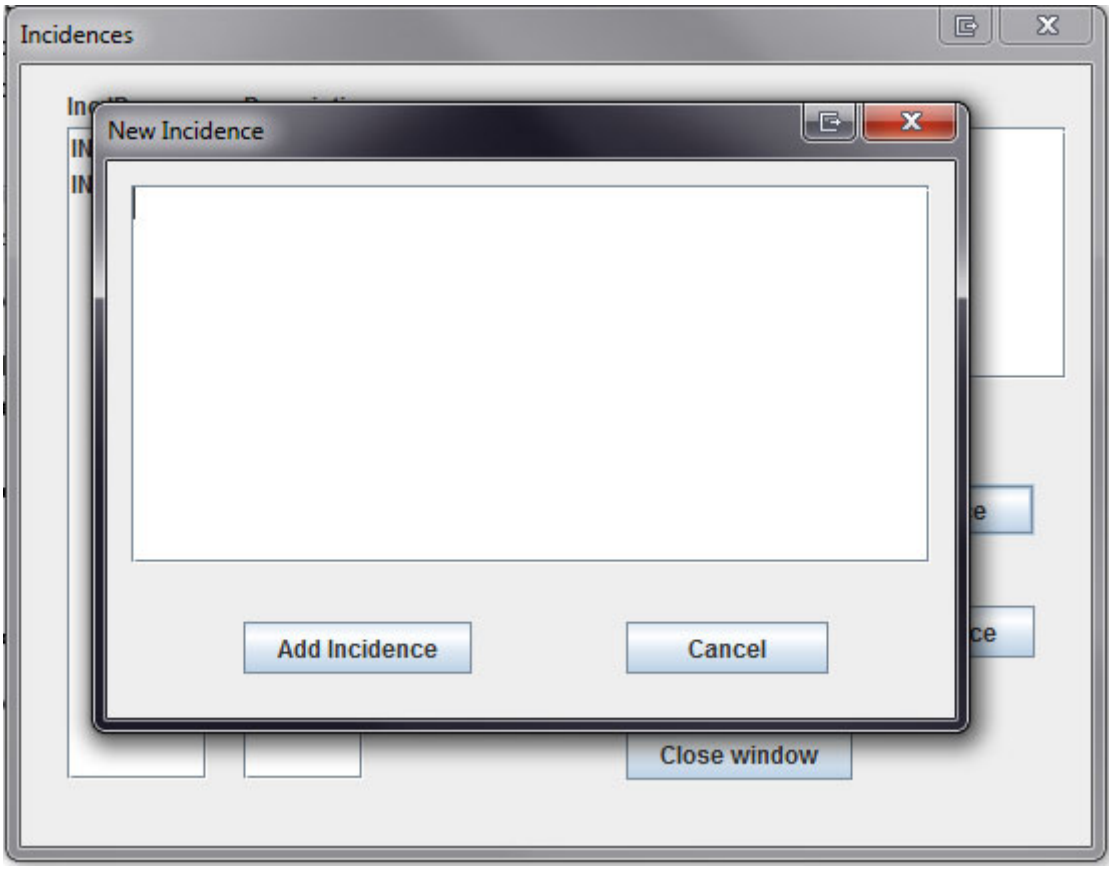

Figura 24: Ventana de nueva Incidencia.

Desde aquí, una vez rellenemos el cuadro de texto (sino mostrará un error de que está vacía al aceptar) podremos guardar los cambios y crear la incidencia con el contenido del cuadro de texto pulsando el botón Aceptar. En cualquier otro caso se cancelará la creación del requisito si pulsamos cancelar o cerramos la ventana.

Si pulsamos el botón “Modificar Incidencia” y habiendo seleccionado previamente una incidencia de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) se mostrará la siguiente ventana:

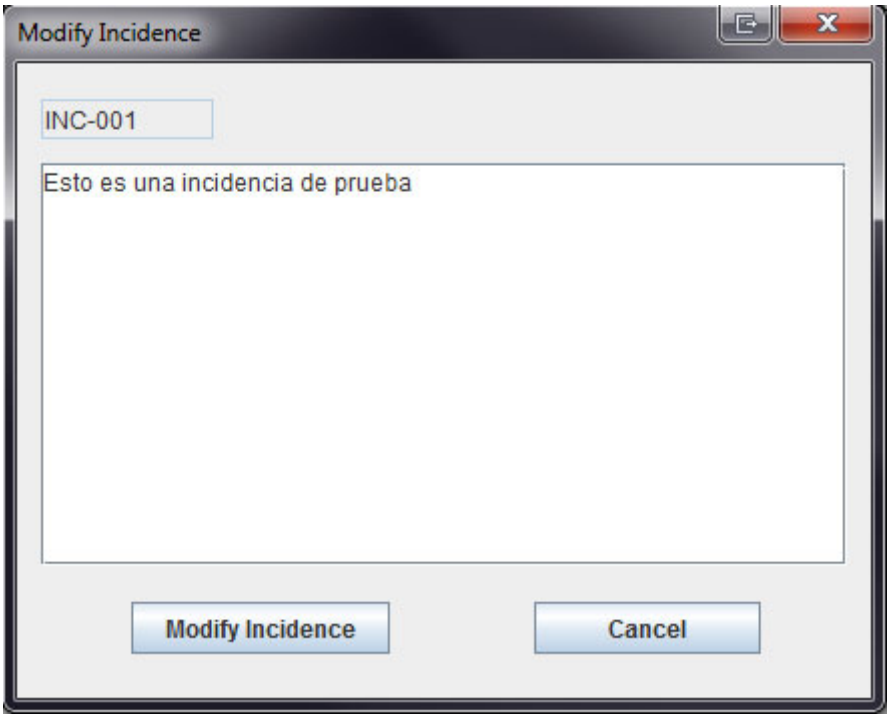

Figura 25: Ventana de modificar Incidencia.

En ella podemos observar la ID de la incidencia seleccionada y un cuadro de texto con la descripción actual con posibilidad de ser modificada. Una vez hayamos realizado los cambios pertinentes podremos guardarlos pulsando el botón Aceptar. En cualquier otro caso, se cancelará la modificación de la incidencia si pulsamos cancelar o cerramos la ventana.

Por último si pulsamos el botón “Borrar Incidencia” y habiendo seleccionado previamente una incidencia de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) se nos mostrará el siguiente cuadro de confirmación:

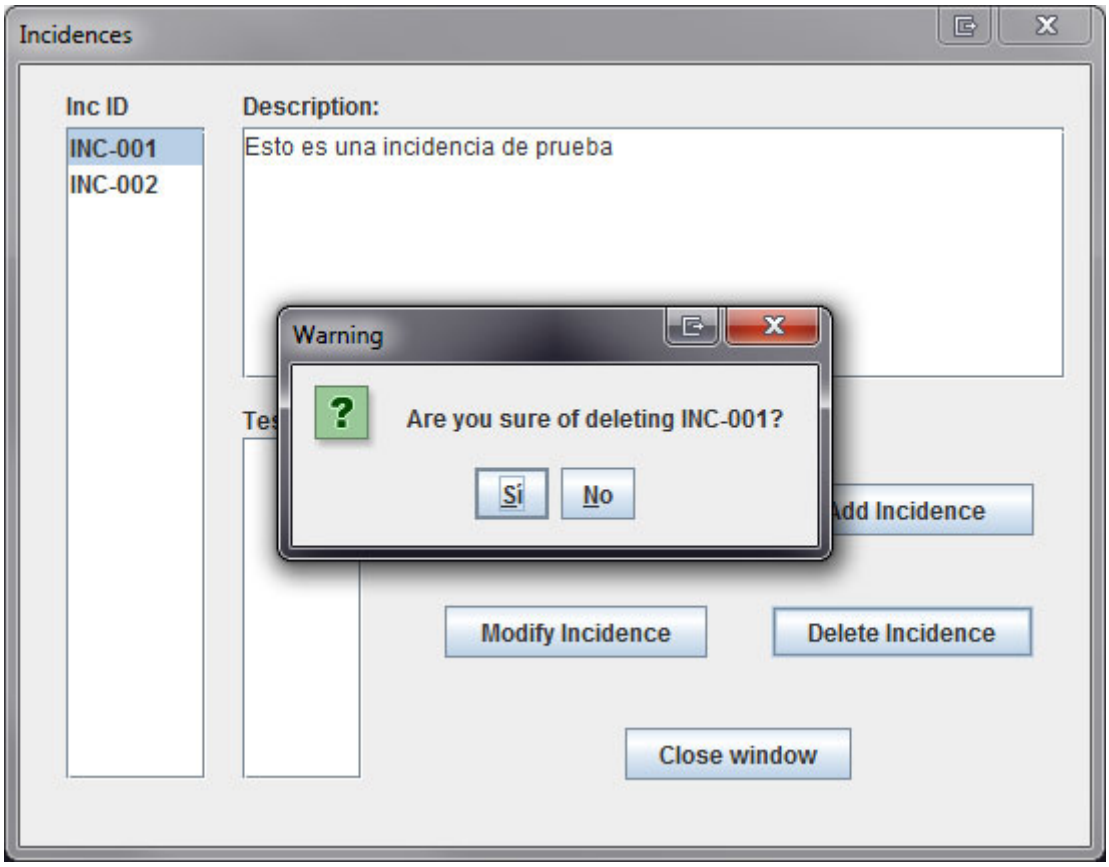

Figura 26: Mensaje de borrado de incidencias.

Desde aquí se podrá borrar la incidencia pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando el mensaje de confirmación.

Ventana Pruebas:

Desde esta ventana podremos administrar todo lo referente a las pruebas, desde creación y modificación hasta asignación y borrado de requisitos e incidencias. Para ello tenemos cada uno de los botones específicos para realizar cada función.

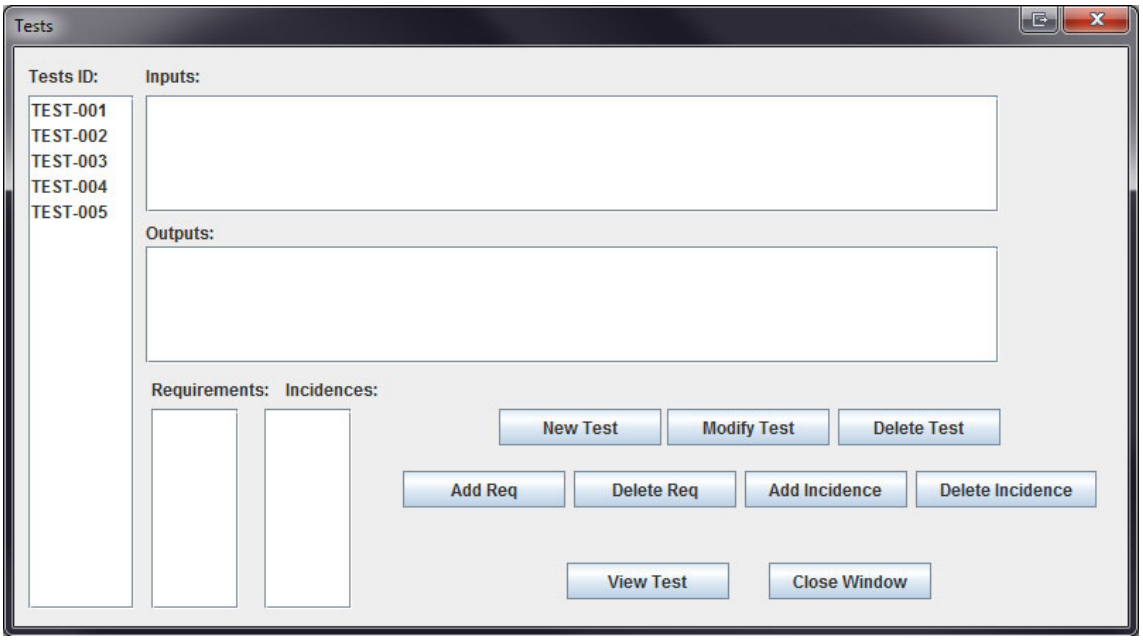

Figura 27: Ventana de Pruebas.

En la lista de la izquierda de la ventana se mostrarán todas las pruebas existentes en la base de datos ordenadas por ID. En la parte derecha encontramos 2 cuadros de texto uno para mostrar la descripción del input y otro para la descripción del output cuando sea requerido. Luego existen otras dos listas más pequeñas en la parte inferior, en estas se mostrarán los requisitos o incidencias (una para cada una) asociadas a la prueba una vez esta se muestre. En caso de quedar en blanco no tendrá ninguno asociado. . Los botones realizarán cada uno su función.

Pulsando el botón “Ver Prueba” y habiendo seleccionado previamente una prueba de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) Se mostrará en la caja de texto correspondiente el input de la prueba, el output de la Prueba y las incidencias y requisitos asociados (en caso de tenerlos). Esta función también puede ser invocada haciendo doble click sobre un requisito de la lista de la izquierda.

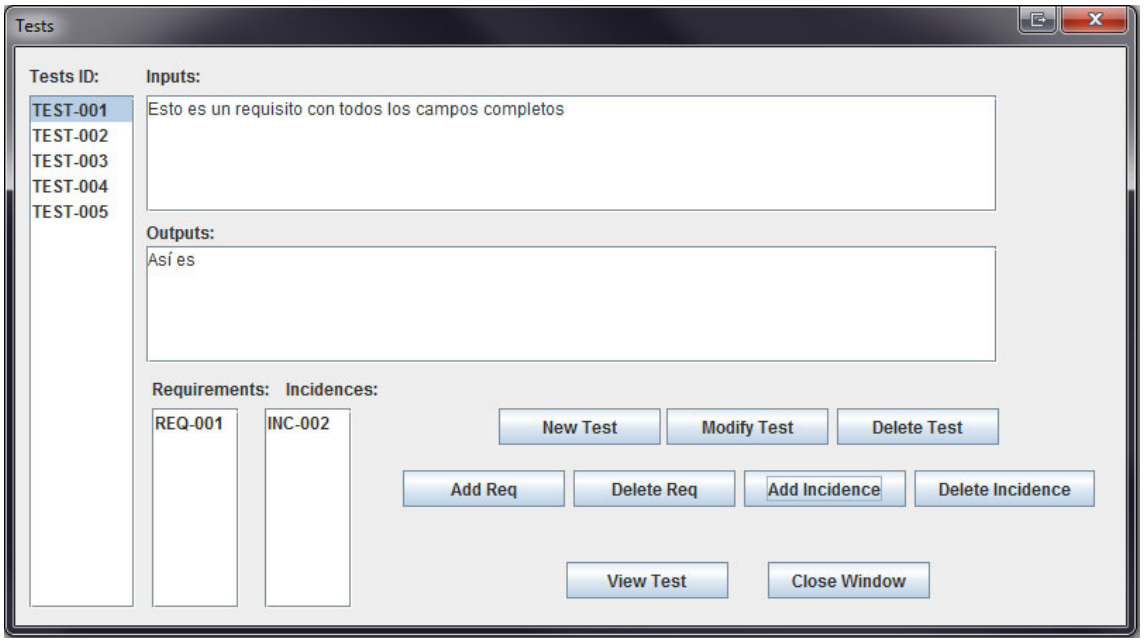

Figura 28: Ventana de Pruebas mostrando todos los campos.

Mediante el botón “Crear Prueba” se podrá añadir una nueva prueba a la base de datos. Para ello y al pulsarlo se nos abrirá la siguiente ventana:

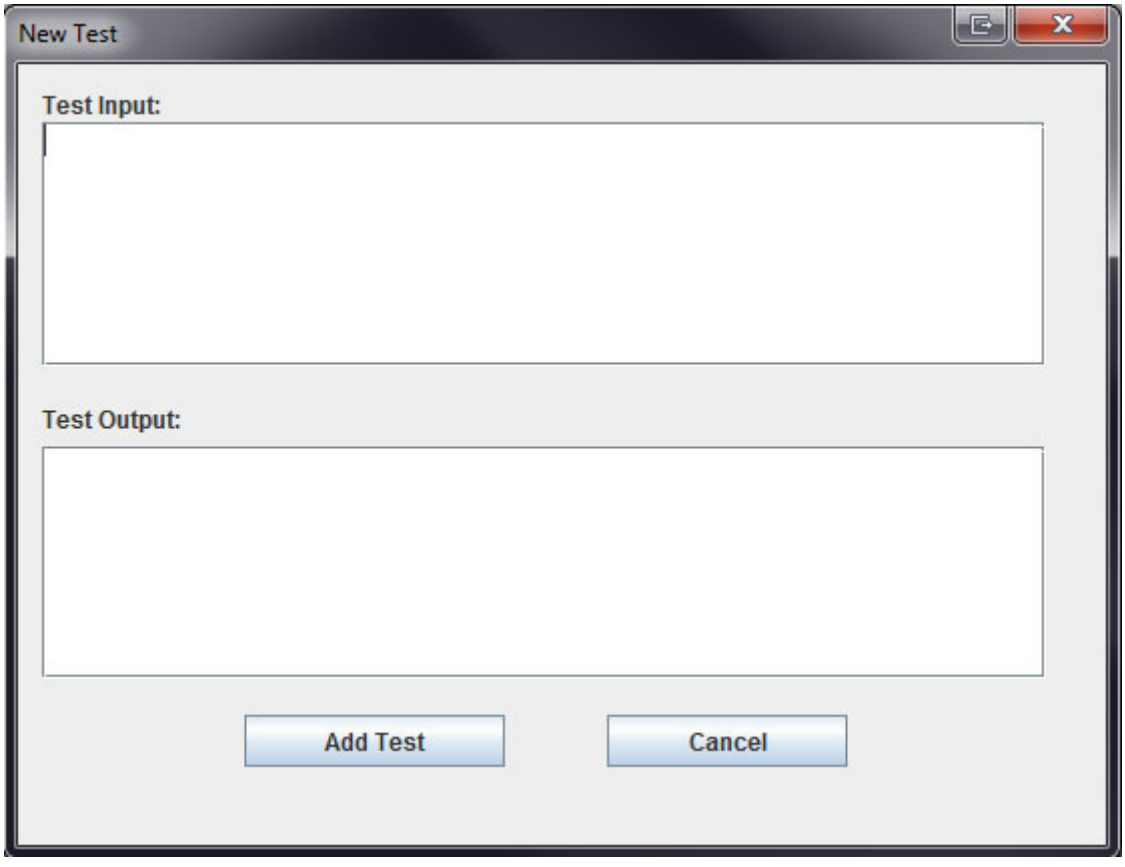

Figura 29: Ventana de nueva prueba.

En esta nueva ventana encontramos dos cuadros de texto modificables en blanco, uno para escribir el input y otro para el output. Se podrá dejar el cuadro de texto del output en

blanco, no obstante el del input deberá rellenarse obligatoriamente (de otra forma mostrará un mensaje de error al aceptar). Podremos guardar los cambios y crear la prueba con el contenido de los cuadros de texto pulsando el botón aceptar. En cualquier otro caso se cancelará la creación del requisito si pulsamos cancelar o cerramos la ventana.

Si pulsamos el botón "Modificar Prueba" y habiendo seleccionado previamente una prueba de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) se permitirá modificar la prueba escogida. Para ello se mostrará la siguiente ventana:

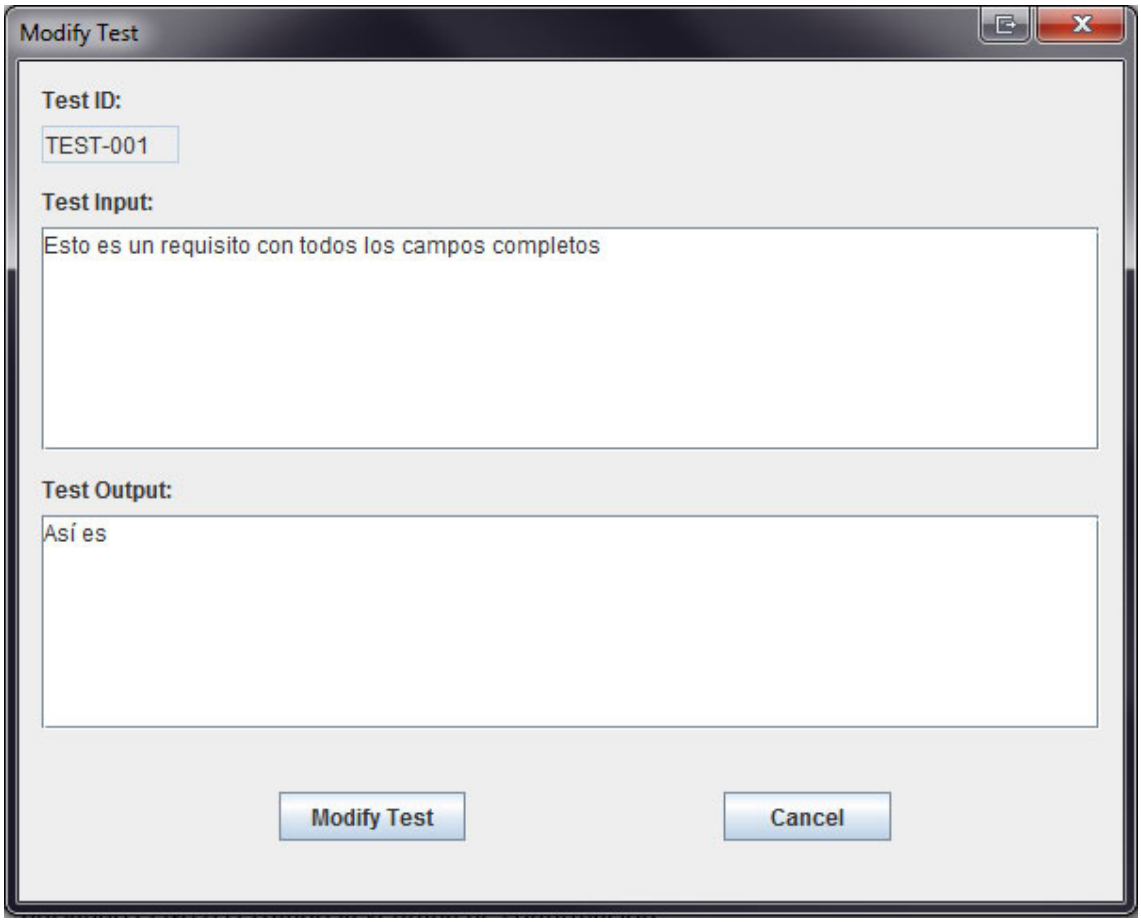

Figura 30: ventana de modificar prueba.

En esta ventana, podemos observar la ID de la prueba seleccionada y dos cuadro de texto con el input y el output actuales y con posibilidad de ser modificados. Una vez hayamos realizado los cambios pertinentes podremos guardarlos pulsando el botón Aceptar (el cuadro input nunca podrá dejarse vacío). En cualquier otro caso, se cancelará la modificación de la prueba si pulsamos cancelar o cerramos la ventana.

El botón "Borrar Prueba" nos permitirá eliminar una prueba (previamente habiendo seleccionado una prueba para borrar). Entonces, se nos mostrará el siguiente cuadro de confirmación:

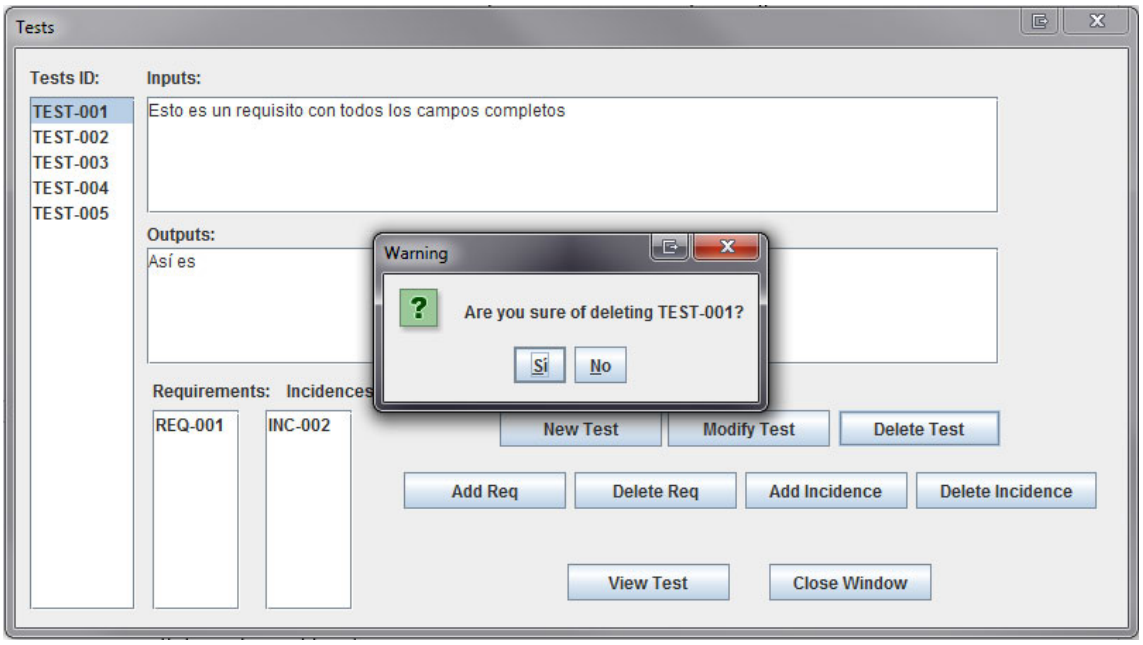

Figura 31: Mensaje de borrado de pruebas.

Desde aquí se podrá borrar la prueba pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando la ventana de confirmación.

Mediante el botón "Asignar Requisito" podremos añadir un requisito que se debe validar para la prueba seleccionada (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna). Para ello al pulsar el botón se nos mostrará la siguiente ventana:

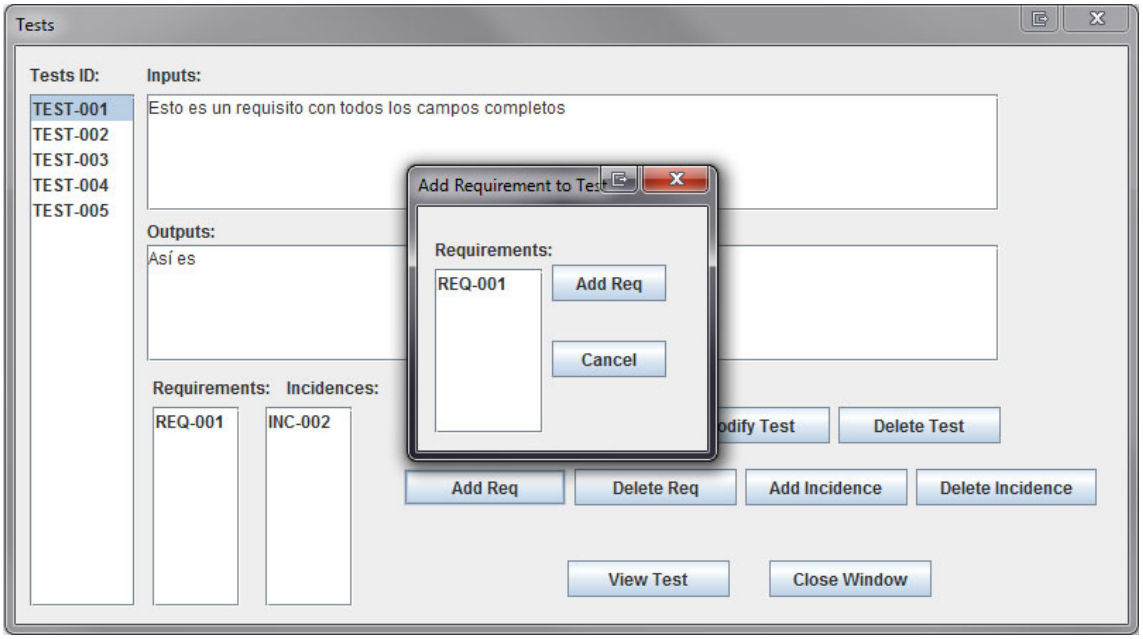

Figura 32: Ventana de añadir requisitos a una prueba.

Desde esta ventana seleccionaremos el requisito de la lista que se muestra en el lado izquierdo. Posteriormente, solo tendremos que pulsar el botón Aceptar si queremos guardar los cambios o pulsar el botón Cancelar o cerrar la nueva ventana en caso de que

queramos cancelar la operación, en cuyo caso no se asociará el requisito para la prueba seleccionada.

Por otro lado el botón “Desasignar Requisito” ejecutará la acción contraria a lo expresado anteriormente. Borrará un requisito asociado a una prueba (en caso de seleccionar un prueba sin requisito asociado mostrará un mensaje de error). Se nos mostrará el siguiente cuadro de confirmación:

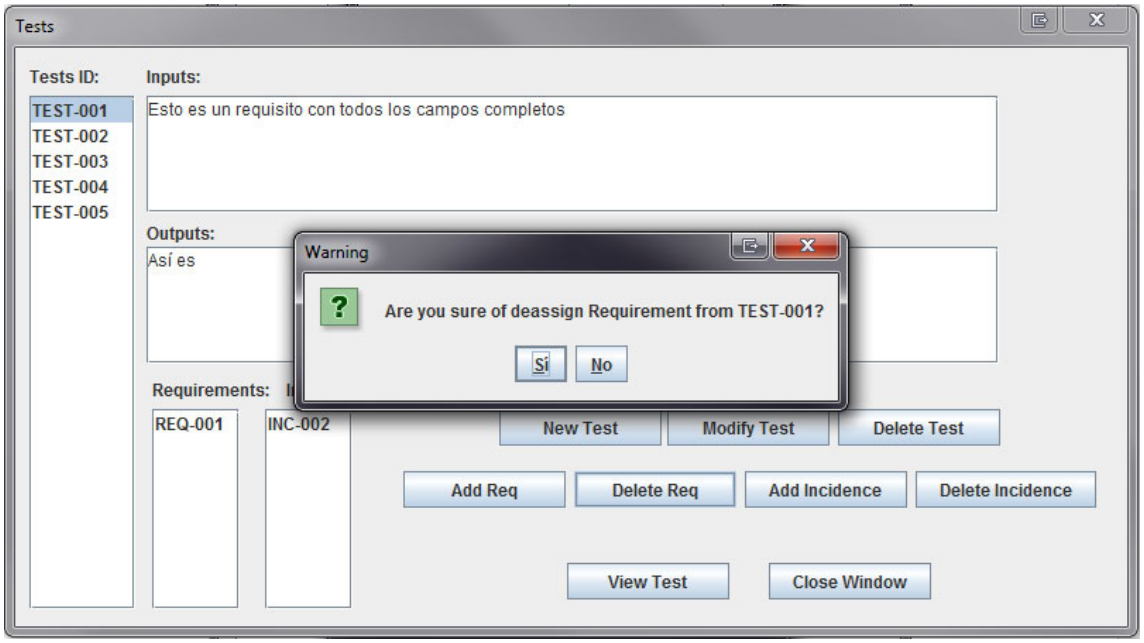

Figura 33: Mensaje de des asignación de requisitos de una prueba.

Desde aquí se podrá borrar el requisito pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando la ventana de confirmación.

Mediante el botón “Asignar Incidencia” Podremos añadir una incidencia para la prueba seleccionada (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna). Para ello al pulsar el botón se nos mostrará la siguiente ventana:

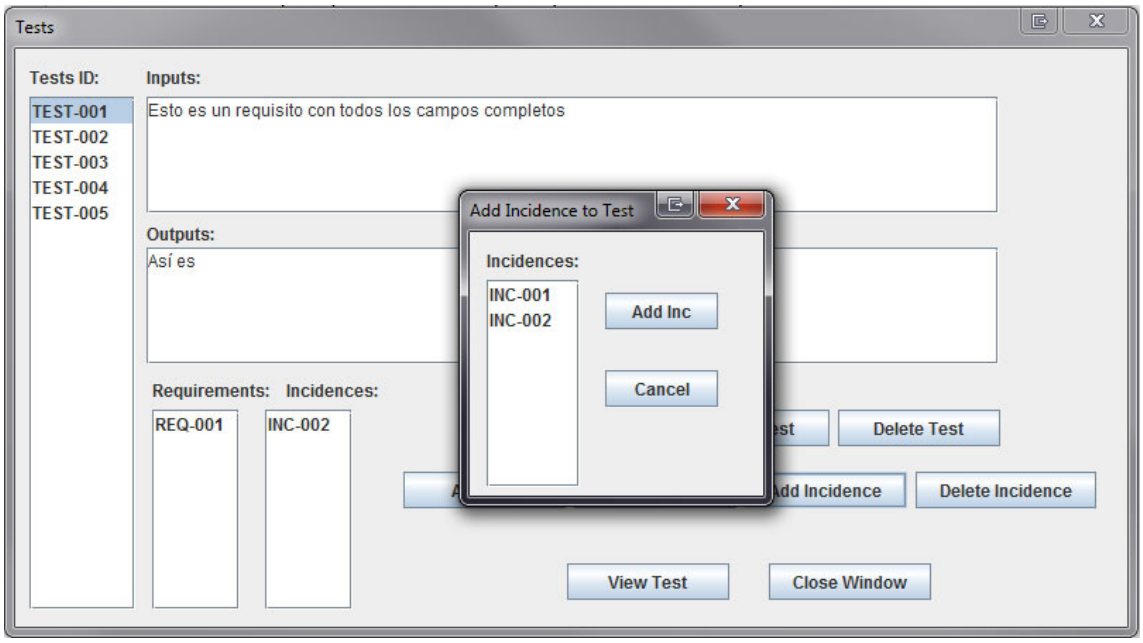

Figura 34: Ventana de añadir incidencias a una prueba.

Desde esta ventana seleccionaremos la incidencia de la lista que se muestra en el lado izquierdo. Posteriormente solo tendremos que pulsar el botón Aceptar si queremos guardar los cambios o pulsar el botón cancelar o cerrar la nueva ventana en caso de que queramos cancelar la operación, en cuyo caso no se asociará la incidencia para la prueba seleccionada.

Por otro lado el botón “Desasignar Incidencia” ejecutará la acción contraria a lo expresado anteriormente. Borrará una incidencia asociada a una prueba (en caso de seleccionar un prueba sin requisito asociado mostrará un mensaje de error). Se nos mostrará el siguiente cuadro de confirmación:

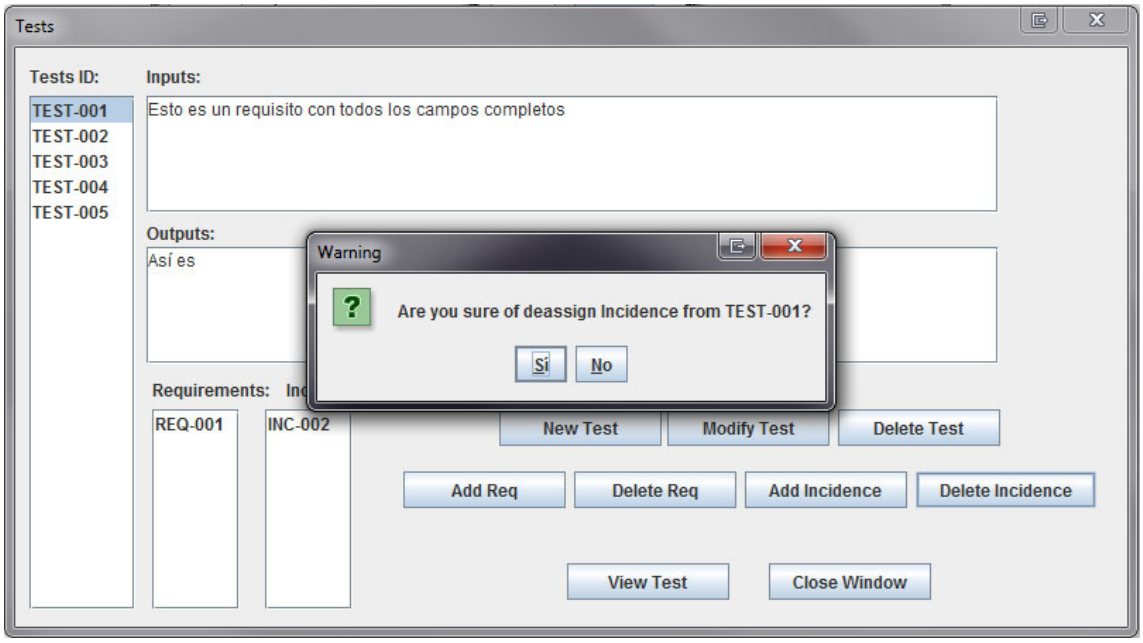

Figura 435: Mensaje de des asignación de incidencias de una prueba.

Desde aquí se podrá borrar la incidencia pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando la ventana de confirmación.

Ventana Resultados:

Desde esta ventana podremos administrar la creación, modificación y borrado de resultados. Para ello tenemos cada uno de los botones específicos para realizar cada función.

Figura 36: Ventana de Resultados.

En la lista de la izquierda aparecerán todas las pruebas existentes en la base de datos. Junto a ésta hay dos pequeñas cajas de texto. En la de más a la izquierda aparecerá la ID de la prueba seleccionada. En la otra caja restante aparecerá el resultado asociado para dicha prueba. Cada botón realizará su función correspondiente.

Pulsando el botón “Ver Resultado” y habiendo seleccionado previamente una prueba de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) Se mostrará en la caja de texto correspondiente el resultado de dicha prueba. Esta función también puede ser invocada haciendo doble click sobre una prueba de la lista de la izquierda.

Mediante el botón “Crear Resultado” podremos añadir un nuevo resultado a la base de datos. Para ello al pulsarlo se nos mostrará la siguiente ventana:

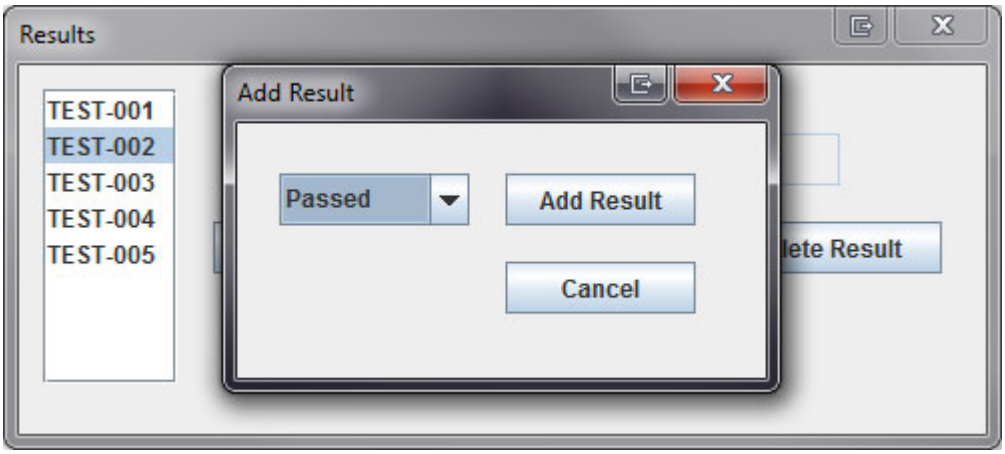

Figura 37: Ventana Crear resultados.

En la lista desplegable seleccionaremos el resultado que queremos asignarle y posteriormente solo tendremos que pulsar el botón Aceptar si queremos guardar los cambios o pulsar el botón cancelar o cerrar la nueva ventana en caso de que queramos cancelar la operación, en cuyo caso no se guardará el resultado para la prueba seleccionada.

Sin pulsamos el botón "Modificar Resultado" y habiendo seleccionado previamente una prueba de la lista (de otra forma nos mostrara un mensaje de error diciendo que no hemos seleccionado ninguna) se permitirá modificar un resultado existente. Para ellos se mostrará la siguiente ventana, similar a la de creación del resultado:

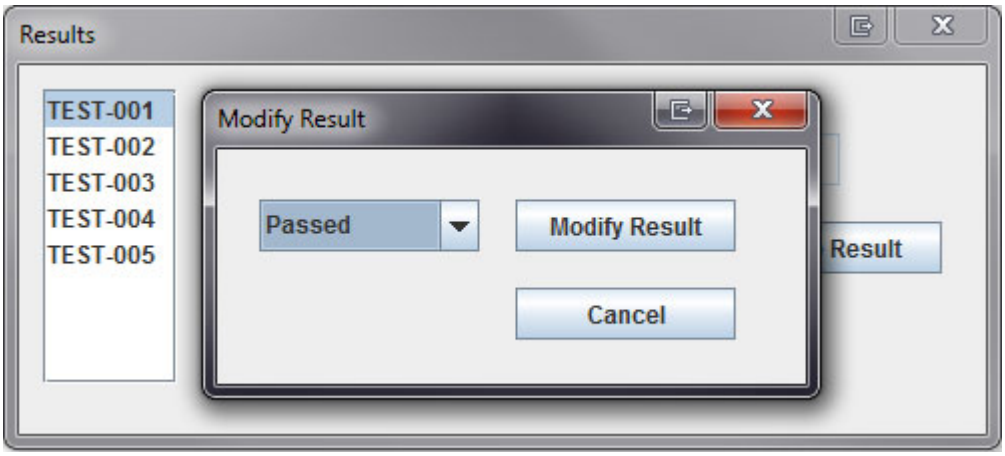

Figura 38: Ventana de modificar resultados.

En la lista desplegable seleccionaremos el resultado que queremos asignarle y posteriormente solo tendremos que pulsar el botón Aceptar si queremos guardar los cambios o pulsar el botón cancelar o cerrar la nueva ventana en caso de que queramos cancelar la operación, en cuyo caso no se guardará el resultado para la prueba seleccionada.

Por último si pulsamos el botón "Borrar Resultado" (previamente habiendo seleccionado una prueba para borrar el resultado que contenga un resultado, sino nos dará un mensaje de error) se nos mostrará el siguiente cuadro de confirmación:

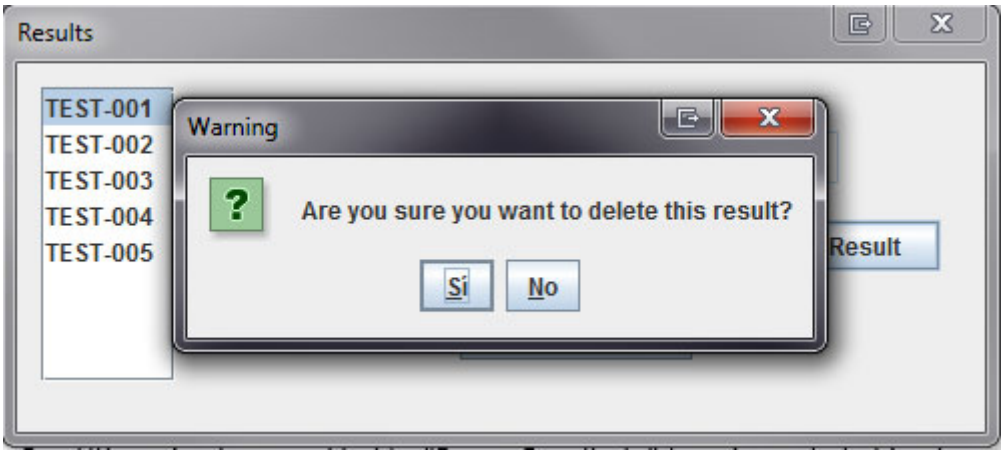

Figura 39: Mensaje de borrado de resultados de una prueba.

Desde aquí se podrá borrar el resultado pulsando el botón Si y se cancelará el borrado pulsando el No o cerrando la ventana de confirmación.

Generalidades de todas las ventanas:

Se puede salir en cualquier momento de cualquier ventana si se pulsa el botón Cancelar presente en todas y cada una de ellas o bien pulsando el botón cerrar propio de las ventanas de Windows en la parte superior derecha de la misma.

8. Pruebas

8.1 Definición y ejecución de pruebas unitarias

A continuación y previa a la descripción de las pruebas individuales ejecutadas, se muestras una tabla resumen con información sobre las mismas.

Casos de Prueba	Ejecución OK		Ejecución No OK		Pendiente		Bloqueado	
	Nº	%	Nº	%	Nº	%	Nº	%
V. Inicial	2	100	0	0	0	0	-	-
V. Pruebas	9	100	0	0	0	0	-	-
V. Reqs	5	100	0	0	0	0	-	-
V. Incidencias	**5**	**100**	**0**	**0**	**0**	**0**	-	-
V. resultados	**4**	**100**	**0**	**0**	**0**	**0**	-	-
TOTAL	**25**	**100**	**0**	**0**	**0**	**0**	-	-

Ventana Inicial:

Crear una nueva base de datos

Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón “Nueva Base de Datos”. Se comprueba que los datos han sido inicializados comprobando los valores de las ventanas: Pruebas, Requisitos, Incidencias y resultados.

Usar la base de datos actual

Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón “Continuar con la Base de Datos”. Se comprueba que los datos no han sido inicializados comprobando los valores de las ventanas: Pruebas, Requisitos, Incidencias y resultados.

Ventana Pruebas:

Crear una nueva prueba

Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón “Continuar con la Base de Datos”. Se pulsa el botón “Pruebas” y a continuación “Nueva Prueba”. Desde la nueva ventana que se abrirá, se introduce el texto de al menos el cuadro de input. Luego se pulsa el botón Aceptar. Se comprueba que la prueba ha sido creada en la ventana Pruebas.

Modificar una prueba existente
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Modificar Prueba”. El texto de la nueva ventana es modificado y se pulsa el botón Aceptar. Se comprueba que la prueba se ha modificado en la ventana “Pruebas”.

Borrar una prueba
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Borrar Prueba”. Desde la ventana de confirmación pulsamos “SI”. Comprobamos que la prueba se ha eliminado de la lista de pruebas.

Asignar un requisito
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Asignar Requisito”. Desde la nueva ventana mostrada se selecciona un requisito existente y pulsamos “Aceptar”. Desde la ventana “Pruebas” comprobamos que se ha asignado correctamente.

Asignar una incidencia
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Asignar Incidencia”. Desde la nueva ventana mostrada se selecciona una incidencia existente y pulsamos “Aceptar”. Desde la ventana “Pruebas” comprobamos que se ha asignado correctamente.

Desasignar un requisito
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Borrar Requisito”. Desde la ventana “Pruebas” comprobamos que se ha desasignado correctamente.

Desasignar una incidencia
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Borrar Incidencia”. Desde la ventana “Pruebas” comprobamos que se ha desasignado correctamente.

Ver una prueba (botón)
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se selecciona la prueba creada anteriormente (o se puede crear una nueva) y pulsamos “Ver Prueba”. La descripción de la prueba y su requisito e incidencia asociados se mostrarán.

Ver una prueba (doble click)
Desde la ventana del menú principal se pulsa el botón “Pruebas” y a continuación se hace doble click sobre una prueba creada anteriormente (o se puede crear una nueva). La descripción de la prueba y su requisito e incidencia asociados se mostrarán.

Ventana Requisitos:

Crear un requisito.
Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón "Continuar con la Base de Datos". Se pulsa el botón "Requisitos" y a continuación "Nuevo Requisito". Desde la nueva ventana que se abrirá, se introduce el texto en el cuadro de Descripción. Luego se pulsa el botón Aceptar. Se comprueba que el requisito ha sido creado en la ventana Requisitos.

Modificar un requisito.
Desde la ventana del menú principal se pulsa el botón "Requisitos" y a continuación se selecciona el Requisito creado anteriormente (o se puede crear uno nuevo) y pulsamos "Modificar Requisito". El texto de la nueva ventana es modificado y se pulsa el botón Aceptar. Se comprueba que el requisito se ha modificado en la ventana "Requisitos".

Borrar un requisito.
Desde la ventana del menú principal se pulsa el botón "Requisitos" y a continuación se selecciona el Requisito creado anteriormente (o se puede crear uno nuevo) y pulsamos "Borrar Requisito". Desde la ventana de confirmación pulsamos "SI". Comprobamos que el Requisito se ha eliminado de la lista de Requisitos.

Ver un requisito (botón)
Desde la ventana del menú principal se pulsa el botón "Requisitos" y a continuación se selecciona el Requisito creado anteriormente (o se puede crear uno nuevo) y pulsamos "Ver Requisito". La descripción del Requisito será mostrado.

Ver un requisito (doble click)
Desde la ventana del menú principal se pulsa el botón "Requisitos" y a continuación se hace doble click sobre un requisito creado anteriormente (o se puede crear uno nuevo). La descripción del requisito se mostrará.

Ventana Incidencias:

Crear una incidencia.
Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón "Continuar con la Base de Datos". Se pulsa el botón "Incidencias" y a continuación "Nueva Incidencia". Desde la nueva ventana que se abrirá, se introduce el texto en el cuadro de Descripción. Luego se pulsa el botón Aceptar. Se comprueba que la incidencia ha sido creada en la ventana Incidencias.

Modificar Una incidencia.
Desde la ventana del menú principal se pulsa el botón "Incidencias" y a continuación se selecciona la Incidencia creada anteriormente (o se puede crear una nueva) y pulsamos "Modificar Incidencia". El texto de la nueva ventana es modificado y se pulsa el botón Aceptar. Se comprueba que la Incidencia se ha modificado en la ventana "Incidencias".

Borrar una incidencia.
Desde la ventana del menú principal se pulsa el botón “Incidencias” y a continuación se selecciona la Incidencia creada anteriormente (o se puede crear una nueva) y pulsamos “Borrar Incidencia”. Desde la ventana de confirmación pulsamos “SI”. Comprobamos que la Incidencia se ha eliminado de la lista de Incidencias.

Ver una incidencia (botón).
Desde la ventana del menú principal se pulsa el botón “Incidencias” y a continuación se selecciona la Incidencia creado anteriormente (o se puede crear una nueva) y pulsamos “Ver Incidencia”. La descripción de la Incidencia será mostrada.

Ver una incidencia (doble click).
Desde la ventana del menú principal se pulsa el botón “Incidencias” y a continuación se hace doble click sobre una Incidencia creada anteriormente (o se puede crear una nueva). La descripción de la Incidencia se mostrará.

Ventana Resultados:

Crear un resultado.
Se abre el programa pulsando doble click en el ejecutable. Desde la ventana Inicial se pulsa el botón “Continuar con la Base de Datos”. Se pulsa el botón “Resultados” y a continuación se selecciona una Prueba de la lista que no contenga ya un resultado. Se pulsará el botón “Nuevo Resultado”. Desde la nueva ventana abierta se selecciona un resultado de la lista desplegable y se pulsa “Aceptar”. Se comprueba que se ha añadido el resultado para esa prueba en la ventana Resultados.

Modificar un resultado.
Desde la ventana Principal se pulsa el botón “Resultados” y a continuación se selecciona una Prueba de la lista que contenga ya un resultado. Se pulsará el botón “Modificar Resultado”. Desde la nueva ventana abierta se selecciona un resultado de la lista desplegable y se pulsa “Aceptar”. Se comprueba que se ha modificado el resultado para esa prueba en la ventana Resultados.

Borrar un resultado.
Desde la ventana Principal se pulsa el botón “Resultados” y a continuación se selecciona una Prueba de la lista que contenga ya un resultado. Se pulsará el botón “Borrar Resultado”. Desde la ventana de confirmación pulsamos “SI”. Comprobamos que el resultado se ha eliminado de la lista de Resultados. Se comprueba que se ha borrado el resultado para esa prueba en la ventana Resultados.

Ver un resultado (doble click).
Desde la ventana del menú principal se pulsa el botón “Resultados” y a continuación se hace doble click sobre una prueba mostrada en la lista. La descripción de la Incidencia se mostrará.

8.2 Definición y ejecución de pruebas de integración y validación

Las pruebas descritas en este apartado de forma resumida se pueden encontrar de una forma más formal y ampliada en los casos de uso anexados al final del documento.

Prueba 1: Se crea un elemento de cada ventana

Prueba 2: Se crea una prueba nueva y se modifica, posteriormente se asignan otros elementos

Prueba 3: A una prueba se le añade un resultado, se modifica el requisito asociado y se borra la incidencia asociada

Prueba 4: se borra un requisito, una incidencia y una prueba existentes

Prueba 5: se modifica el resultado de una prueba y su descripción. Se le borra el resultado asignado posteriormente.

8.3 Tabla de trazabilidad de los requisitos.

Id Req	Id Caso de Prueba
Requisito 1	No funcionales, probado en todas las pruebas.
Requisito 2	No funcionales, probado en todas las pruebas.
Requisito 3	No funcionales, probado en todas las pruebas.
Requisito 4	No funcionales, probado en todas las pruebas.
Requisito 5	No funcionales, probado en todas las pruebas.
Requisito 6	No funcionales, probado en todas las pruebas.
Requisito 7	No funcionales, probado en todas las pruebas.
Requisito 8	No funcionales, probado en todas las pruebas.
Requisito 9	No funcionales, probado en todas las pruebas.
Requisito 10	No funcionales, probado en todas las pruebas.
Requisito 11	No funcionales, probado en todas las pruebas.
Requisito 12	No funcionales, probado en todas las pruebas.
Requisito 13	prueba 1
Requisito 14	prueba 1
Requisito 15	prueba 1
Requisito 16	prueba 1
Requisito 17	prueba 1
Requisito 18	prueba 1
Requisito 19	prueba 1
Requisito 20	prueba 1
Requisito 21	prueba 1
Requisito 22	prueba 1

Requisito 23	prueba 1
Requisito 24	prueba 1
Requisito 25	prueba 1
Requisito 26	prueba 1
Requisito 27	prueba 3
Requisito 28	prueba 1
Requisito 29	prueba 1,2,3 y 4
Requisito 30	prueba 1,2,3 y 4
Requisito 31	prueba 1,2,3 y 4
Requisito 32	prueba 1,2,3 y 4
Requisito 33	prueba 1,2,3 y 4
Requisito 34	prueba 1,2,3 y 4
Requisito 35	prueba 3 y 5
Requisito 36	prueba 3 y 5
Requisito 37	prueba 3 y 5
Requisito 38	prueba 3 y 5
Requisito 39	prueba 3 y 5
Requisito 40	Todas las pruebas
Requisito 41	Todas las pruebas
Requisito 42	Todas las pruebas
Requisito 43	Todas las pruebas
Requisito 44	Todas las pruebas
Requisito 45	prueba 1,2 y 4
Requisito 46	prueba 1,2 y 4
Requisito 47	prueba 1,2 y 4
Requisito 48	prueba 1,2 y 4
Requisito 49	prueba 1,2 y 4
Requisito 50	prueba 1,2 y 4
Requisito 51	prueba 1
Requisito 52	prueba 1
Requisito 53	prueba 1
Requisito 54	prueba 1
Requisito 55	prueba 3
Requisito 56	prueba 3
Requisito 57	prueba 3
Requisito 58	prueba 1 y 2
Requisito 59	prueba 1 y 2
Requisito 60	prueba 1 y 2
Requisito 61	prueba 1 y 2
Requisito 62	prueba 1
Requisito 63	prueba 1
Requisito 64	prueba 1
Requisito 65	prueba 1
Requisito 66	prueba 3
Requisito 67	prueba 3
Requisito 68	prueba 3

Requisito 69	prueba 3
Requisito 70	prueba 5
Requisito 71	prueba 5
Requisito 72	prueba 5
Requisito 73	prueba 1 y 2
Requisito 74	prueba 1 y 2
Requisito 75	prueba 1 y 2
Requisito 76	prueba 1 y 2
Requisito 77	prueba 3
Requisito 78	prueba 3
Requisito 79	prueba 3
Requisito 80	prueba 3
Requisito 81	prueba 2
Requisito 82	prueba 2
Requisito 83	prueba 2
Requisito 84	prueba 2
Requisito 85	prueba 2
Requisito 86	prueba 2
Requisito 87	prueba 2
Requisito 88	prueba 2

9. Conclusiones y propuestas de mejora

9.1 Resultados

Llegados a este punto, hay muchas conclusiones que se me ocurren. Todas ellas son experiencias de lo obtenido mientras tenía lugar la realización de este proyecto. Hace unos meses cuando comencé con el planteamiento de qué era exactamente lo que me proponía a realizar y cómo lo quería exactamente, no llegue a imaginarme lo satisfecho que me sentiría con el resultado.

Cuando comencé el proyecto parecía algo difícil de conseguir y lejano en el tiempo. Realmente me ha llevado más tiempo del que había supuesto por temas de trabajo y retrasos varios, lo cual me hace sentirme un poco menos contento pero no por ello menos satisfecho. Una vez contemplado el trabajo realizado y el producto final desarrollado no podría estar más satisfecho con lo que he desarrollado, aprendido y también sufrido. Ya que si no hubiera supuesto un desafío no me habría sentido de la misma manera.

Con este proyecto he aprendido a enfrentarme a problemas y contratiempos que podrían suponer un trabajo real. He podido emplear todos los conocimientos aprendidos durante estos 4 años de carrera asignatura tras asignatura y además, de aprender otros nuevos que han sido muy interesantes y que me servirán en un futuro ya sea para temas laborales o por satisfacción propia y ansia de conocimiento. He sido capaz de usar nuevas herramientas de desarrollo así como conocimientos emergentes y bastante actuales. Lo cual me ha llevado al límite a la hora de resolver dudas, buscando durante días información, blogs y foros para resolverlas. Me ha llevado a pedir ayuda por correo de personas que dominaban el tema y ha sido no solo satisfactorio sino además muy positivo, ya que he aprendido que hay cosas que a veces uno solo no es capaz de resolver por su cuenta y hay que depender de los demás, como en una empresa, en la que puedes depender de tus compañeros. Cosas que parecían muy fáciles se han convertido en las más difíciles y viceversa, aquellas que desconocía han resultado ser muy sencillas de entender y otras que creía dominar muy complejas para desarrollar.

Una vez completado este proyecto y debido a nuevos conocimientos y posibilidades de mejora del programa me ha hecho darme cuenta de todo lo que aún puede incluirse al mismo y será detallado en un apartado posterior a este. Con esto no quiero decir que mi trabajo esté incompleto, si no que puede servir incluso a otra persona interesada en el tema a investigar y mejorar de algún modo dicha herramienta, para que pueda cumplir su función aún mejor.

Para finalizar, con esta memoria consigo graduarme como Ingeniero, algo que no pensaba que acabaría en tan solo 4 años. Además toda esta información y conocimientos me sirven para mi trabajo actual como Ingeniero de Pruebas. Quiero agradecer todo el apoyo recibido por parte de la tutora, los profesores y los compañeros durante estos 4 años, que también han ayudado a mi mejora como persona.

9.2 Conclusiones

Al ser un área de inherente ejercicio profesional, la parte correspondiente de la Ingeniería de Pruebas suele enfocarse desde un punto de vista teórico más que práctico. Hay muchas herramientas para creación de pruebas y de ayuda para los ingenieros de pruebas, pero la mayoría son de pago o hechas a medida para grandes empresas que necesitan dicho software. Con este proyecto , se han propuesto dos contribuciones a saber: una metodología y una herramienta software denominada NovaTest, que sirve de apoyo para el aprendizaje y la práctica de las actividades propias derivadas del procesos realizados en entornos o casos de uso reales tanto a estudiantes de carreras de Ingeniería de Software y afines, a la comunidad universitaria y académica, así como también a los Ingenieros de Pruebas Junior o Ingenieros recién egresados en este campo de la Ingeniería del Software. Con estas dos contribuciones se espera minimizar el impacto que los estudiantes recién egresados de carreras como Ingeniería de Software y afines, sienten cuando son vinculados a trabajos en el mercado laboral como Ingenieros de Pruebas. Con estas dos contribuciones, se ha pretendido de una forma sistemática y fácil de seguir, brindar a los una visión general de lo que sería la Ingeniería de pruebas de una forma similar a como podrías encontrar en una empresa dedicada a ello. Es este aspecto casi lo que más me agrada del proyecto, ya que no solo me vale para logros personales sino que además otras personas podrán beneficiarse de mi trabajo y podrá servirles para mejorar como ingenieros Informáticos especializados en el área de pruebas.

La metodología que se propone como contribución en este proyecto de fin de grado es novedosa y creativa. Está basada en mi experiencia personal como Ingeniero de Pruebas en un entorno laboral real y sirve como base para que tanto alumnos como Ingenieros Prueba Junior aprendan de forma rigurosa los principales conceptos y conocimientos necesarios para ejercer como profesionales dedicados a la garantía de calidad de un software que debe ser implantado en un entorno real.

Por último, el desarrollo de NovaTest me ha permitido poner en práctica la metodología que he propuesto para la Ingeniería de Pruebas. De esta forma, dentro del mismo desarrollo de NovaTest he seguido mi metodología de tal forma que en una manera práctica, puedo afirmar que resulta de gran ayuda al proceso de verificación y validación de un software.

9.3 Propuestas de mejora

En esta sección se procederá a explicar algunas propuestas que no se han incluido en el alcance de este proyecto para no alargarlo durante más tiempo pero que se deja constancia de ellas para futuras posibilidades de que alguien lo use como proyecto. Entre las posibles mejoras cabe destacar las siguientes:

- Una de las principales carencias de la herramienta software es la falta de datos distribuidos, es decir, para poder seguir las pruebas de un software determinado es necesario ejecutar la herramienta en el mismo ordenador constantemente ya que es en el único sitio donde se guardan, por ello se propone crear una base de datos distribuida, donde los datos se guarden y sean accesible desde varios ordenadores. Para ello habría que meter algún tipo de sistema distribuido como una página web o similar.

- Otro problema menos grave es que para poder usar diferentes bases de datos, en caso necesario, no hay una forma intuitiva de cambiarla. Debes ir al .bin de la carpeta MongoDB y modificarlo para especificar la ruta donde quieres que se guarde la base de datos nueva. Esto podría mejorarse dando una opción para cambiarla desde la herramienta.

- Otro aspecto importante y que está mencionado en la metodología propuesta pero no se ha implementado en la herramienta son los estados de las incidencias y los resultados de las pruebas. Es decir, las anomalías actualmente solo pueden estar creadas o borradas. Sin embargo se debería poder transicionar las incidencias para no tener que borrarlas al corregirlas. Estos estados podrían incluir: abierta, cerrada, pendiente y corregida, por ejemplo y así sería más fácil y mejor para llevar la cuenta de cuantas anomalías abiertas/cerradas hay en la herramienta. Esto se haría incluyendo un nuevo campo de "estado" en la colección de las anomalías para cada documento nuevo creado. En caso de los resultados podrían existir otros diversos como: borrado, fuera de scope, etc.

- Referente a lo anterior, también podría añadirse dos tipos distintos para los resultados con una nueva variable. De este modo unos resultados tendrían carácter interno (sin haber sido ejecutados y demostrados frente al cliente) y otros tendrían un carácter formal, es decir, aquellos que serían válidos para el cliente. Para ello simplemente habría que añadir un campo más a los resultados por ejemplo.

- Actualmente en la herramienta, a una prueba se le puede asignar una y solo una incidencia y únicamente un requisito. Esto no debería ser así, debería ser posible, para que sea más fácil y llevadero ya que una misma prueba puede abarcar la prueba de varios requisitos, asignar varios requisitos o incluso varias incidencias a una misma prueba. Esta es otra propuesta de mejora para el software. Debería cambiarse los campos de pruebas de requisitos e incidencias a un array y que pueda incluir más de un requisito o incidencia sin que estén repetidos.

- Otra mejora que podría ser me mucha utilidad es añadir alguna funcionalidad que haga un resumen global del estado de la base de datos, incluyendo pruebas que hay con sus requisitos, resultados e incidencias asociadas, número de incidencias (cuantas cerradas y cuantas abiertas si se incluye el cambio de estado), cuantos resultados pasados/fallados hay en el sistema, etc.

- Como última propuesta de mejora sería la posibilidad de encontrar los resultados en forma de tabla en lugar de seleccionar las pruebas una por una y ver qué resultado tienen. Sería una forma más fácil de verlo, ya que de un solo vistazo puedes hacerte una

idea de cuantas pruebas están con resultado y cuáles no, y cuales fallan y cuales pasan. Simplificaría mucho esta tarea y sería muy interesante tenerlo disponible.

Como aclaración a estas propuestas, no están en el software porque no se tomaron en cuenta en el análisis que se hizo inicialmente, se han ido descubriendo a medida que el desarrollo iba avanzando y no se han podido incluir por falta de tiempo. Estas propuestas serían simples añadidos que harían más fácil e intuitiva la herramienta, no obstante, toda la funcionalidad básica necesaria para poder probar un software sencillo está incluido en la misma. Es decir, es posible usarla de forma real para software sencillo y como herramienta de prácticas para la universidad que es donde está pensada su implementación.

Con estas propuestas termina esta memoria del proyecto NOVATests y doy por finalizado este proyecto que tanto me ha costado completar y del que sin embargo estoy orgulloso.

10. Bibliografía

[1] Introduction to Software Testing. Paul Ammann and Jeff Offut. Cambridge University Press. 2008.

[2] Universidad Nacional de La Plata. Facultad de Informática. Trabajo Final Integrador de la Especialización en Ingeniería de Software. Estado del arte y tendencias en Test-Driven Development. Autor: Moisés Carlos Fontela

[3] SEMINARIO DE VERIFICACIÓN & VALIDACIÓN Y PRUEBAS UNITARIAS. GMV – DEPARTAMENTO OSV. Autor: Ana Isabel Rodriguez.

[4] http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052010000100004

[5] http://www.fdi.ucm.es/profesor/gmendez/docs/is0809/03-requisitos.pdf

[6] http://oa.upm.es/10750/1/PFC_ALEJANDRO_GONZALEZ_SANTIUESTE__INFORME.pdf

[7] http://www.pmoinformatica.com/2014/05/plan-de-pruebas-de-software.html

[8] http://es.slideshare.net/choselin/plan-de-pruebas-15563690

[9] https://riunet.upv.es/bitstream/handle/10251/8634/Memoria%20proyecto.pdf

[10] http://www.issi.uned.es/ITI/isw/PFC.pdf

[11][Abbott, 1986]. J. Abbott, Software Testing Techniques, NCC Publications, Manchester, England, United Kingdom, 1986.

[12][Beizer, 1983]. B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, New York, New York, 1983.

[13]Software testing and Continuos Qiality Improvement. Third Edition. William Lewis, CRCPress, 2009

[14]Software Testing and Quality Assurance. Theory and Practice. Kshitrasagar Naik y Priyadarshi Tripahty. Willey, 2008

11. Anexo: Documentación de los Casos de Uso

Caso de Uso	Prueba 1	
Descripción	Se crea un elemento de cada ventana	
Actor Iniciador	Usuario	
Resumen	Cualquiera podrá crear elementos nuevos en la herramienta	
Flujo de eventos	**Interacción del usuario**	**Respuestas del Sistema**
	1. Desde la ventana pruebas mediante el botón nueva prueba y rellenando el cuadro de texto (input al menos output es opcional) mostrado se creará una nueva prueba y se comprobará su correcta creación en la ventana de pruebas. 2. Desde la ventana requisitos mediante el botón nuevo requisito y rellenando el cuadro de texto mostrado se creará un nuevo requisito y se comprobará su correcta creación en la ventana de requisitos. 3. Desde la ventana incidencias mediante el botón nueva incidencia y rellenando el cuadro de texto mostrado se creará una nueva incidencia y se comprobará su correcta creación en la ventana de incidencias.	1. La nueva prueba es creada y almacenada en la base de datos con los campos rellenos especificados por el usuario. 2. El nuevo requisito es creado y almacenado en la base de datos con los campos rellenos especificados por el usuario. 3. La nueva incidencia es creada y almacenada en la base de datos con los campos rellenos especificados por el usuario.
Extensiones síncronas	En cualquier ventana la operación puede cancelarse si se pulsa el botón cancelar de las ventanas de creación.	
Extensiones Asíncronas	-	

Caso de Uso	Prueba 2	
Descripción	Se crea una prueba nueva y se modifica, posteriormente se asignan otros elementos	
Actor Iniciador	Usuario	
Resumen	Cualquiera podrá crear elementos nuevos en la herramienta además de modificarla y asignar requisitos e incidencias cuando interese.	
Flujo de eventos	**Interacción del usuario**	**Respuestas del Sistema**
	1. Desde la ventana pruebas mediante el botón nueva prueba y rellenando el cuadro de texto (input al menos output es opcional) mostrado se creará una nueva prueba y se comprobará su correcta creación en la ventana de pruebas. 2. Desde la ventana pruebas mediante el botón modificar prueba y cambiando el cuadro de texto original mostrado se modificará la prueba y se comprobará su correcta modificación en la ventana de pruebas. 3. Desde la ventana pruebas mediante los botones añadir requisito y añadir incidencia y seleccionando un requisito e incidencia existentes y pulsando el botón aceptar se asignaran a la prueba seleccionada y se comprobará su correcta creación en la ventana de pruebas.	1. La nueva prueba es creada y almacenada en la base de datos con los campos rellenos especificados por el usuario. 2. La prueba es modificada y almacenada en la base de datos con los campos rellenos especificados por el usuario. 3. La prueba seleccionada es rellenada con el requisito y la incidencia seleccionados por el usuario.
Extensiones síncronas	En cualquier ventana la operación puede cancelarse si se pulsa el botón cancelar de las ventanas de creación, modificación y	

	asignación.
Extensiones Asíncronas	-

Caso de Uso	Prueba 3	
Descripción	A una prueba se le añade un resultado, se modifica el requisito asociado y se borra la incidencia asociada	
Actor Iniciador	Usuario	
Resumen	Los resultados se podrán añadir libremente desde su ventana correspondiente y se podrá modificar y desasignar de una prueba sus elementos.	
Flujo de eventos	**Interacción del usuario**	**Respuestas del Sistema**
	1. Desde la ventana resultados mediante el botón añadir resultado y habiendo seleccionado una prueba de la lista, se creará un nuevo resultado asociado a la prueba y se comprobará su correcta creación en la ventana de resultados.	1. El nuevo resultado es creado y almacenado en la base de datos.
	2. Desde la ventana requisitos mediante el botón modificar requisito y cambiando el cuadro de texto original mostrado se modificará el requisito y se comprobará su correcta modificación en la ventana de requisitos.	2. El requisito es modificado y almacenado en la base de datos con los campos rellenos especificados por el usuario.
	3. Desde la ventana pruebas mediante el botón borrar incidencia y seleccionando una prueba existente, aceptando los cambios en la confirmación se borrará de la prueba seleccionada y se comprobará su correcta	3. La prueba seleccionada es actualizada con el borrado de la incidencia seleccionados por el usuario.

	creación en la ventana de pruebas.	
Extensiones síncronas	En cualquier ventana la operación puede cancelarse si se pulsa el botón cancelar de las ventanas de creación, modificación y des asignación.	
Extensiones Asíncronas	-	

Caso de Uso	Prueba 4	
Descripción	se borra un requisito, una incidencia y una prueba existentes	
Actor Iniciador	Usuario	
Resumen	Cualquiera podrá borrar elementos existentes en la herramienta	
Flujo de eventos	**Interacción del usuario**	**Respuestas del Sistema**
	1. Desde la ventana pruebas mediante el botón borrar prueba y confirmando el borrado la prueba se eliminará de la base de datos y se comprobará su correcta eliminación en la ventana de pruebas.	1. La prueba especificada por el usuario es borrada y eliminada en la base de datos.
	2. Desde la ventana requisitos mediante el botón borrar requisito y confirmando el borrado del requisito se eliminará de la base de datos y se comprobará su correcta eliminación en la ventana de requisitos.	2. El requisito especificado por el usuario es borrado y eliminado en la base de datos.
	3. Desde la ventana incidencias mediante el botón borrar incidencia y confirmando el borrado de la incidencia se eliminará de la base de datos y se comprobará su correcta eliminación en la ventana de	3. La incidencia especificada por el usuario es borrada y eliminada en la base de datos.

	incidencias.	
Extensiones síncronas	En cualquier ventana la operación puede cancelarse si se pulsa el botón cancelar.	
Extensiones Asíncronas	-	

Caso de Uso	Prueba 5	
Descripción	Se modifica el resultado de una prueba. Se le borra el resultado asignado posteriormente.	
Actor Iniciador	Usuario	
Resumen	Cualquiera podrá borrar resultados existentes de las pruebas.	
Flujo de eventos	**Interacción del usuario**	**Respuestas del Sistema**
	1. Desde la ventana resultados mediante el botón modificar resultado y seleccionando el resultado deseado se modificará el resultado actual. Para ello se pulsará el botón aceptar y se comprobará el cambio realizado en la ventana de resultados. 2. Desde la ventana resultados mediante el botón borrar resultado, habiendo seleccionado la prueba del apartado anterior, y confirmando el cambio de la ventana emergente se borrará el resultado asociado. Esto deberá comprobarse en la ventana resultados.	1. El resultado es modificado para la prueba seleccionada 2. El resultado de la prueba escogida será eliminado de la base de datos.
Extensiones síncronas	En cualquier ventana la operación puede cancelarse si se pulsa el botón cancelar.	
Extensiones Asíncronas	-	